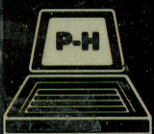


Animation, Games, and Sound For the TI 99/4A

Tony Fabbri



PRENTICE-HALL PERSONAL COMPUTING SERIES

ANIMATION, GAMES, AND SOUND FOR THE TI 99/4A

DISCARD

Tony Fabbri

DISCARD

Prentice-Hall, Inc.

Englewood Cliffs, New Jersey 07632

Fabbri, Tony. (date)

Animation, games, and sound for the TI 99/4A

(Prentice-Hall personal computing series)

Includes index.

1. Computer graphics. 2. Computer animation. 3. Computer games. 4. Computer music. 5. TI 99/4A (Computer)

--Programming. I. Title. II. Series.

T385.F327 1984 001.64'43 84-6965

ISBN 0-13-037227-7

Editorial/production supervision: **Karen Skrable Fortgang**

Interior design: **Kathryn Gollin Marshak**

Cover design: **Jeannette Jacobs**

Manufacturing buyer: **Gordon Osbourne**

Page layout: **Marie Dobish**

TI 99/4A is a trademark of Texas Instruments Incorporated. REF

T
385
.F327
1984

©1984 by **Prentice-Hall, Inc.**, Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-037227-7 01

Prentice-Hall International, Inc., *London*
Prentice-Hall of Australia Pty, Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

CONTENTS

	EDITOR'S FOREWORD	xiii
	PREFACE	xv
Chapter 1	INTRODUCTION	1
Chapter 2	SIMPLE PRINTING	2
	Program 2-1 Print Animal Names 13	
	Program 2-2 Print DOG 20 Times 15	
	Program 2-3 Print Your Name Ten Times 15	
	Program 2-4 Your Name in Columns 16	
	Program 2-5 Your Name Filling the Screen 16	
	Program 2-6 Your Name Fancy 17	
	Program 2-7 Your Name in a Diagonal Pattern 19	
	Program 2-8 Friend's Name in a Diagonal Pattern 20	
	Program 2-9 Two Names in a Crossing Pattern 20	
Chapter 3	ROCKETS FLY, BLAST OFF, AND CHASE ONE ANOTHER	22
	Program 3-1 Rocket 22	
	Program 3-2 Rocket Flying Fast 23	
	Program 3-3 Enemy Rocket Chasing Two Rockets, with Smoke 24	
	Program 3-4 Alien Spacecraft Flying, with Smoke 25	

Chapter 4**ENTERING NAMES AND DRAWING GREETING CARDS****27**

Program 4-1	Printing Your Name 20 Times	28
Program 4-2	Your Name Moving Right	28
Program 4-3	Input Names in a Crossing Pattern	29
Program 4-4	Input Names Around the Screen	30
Program 4-5	Compacted Names All Over	31
Program 4-6	Valentine's Day Card	31
Program 4-7	Christmas Card	32
Program 4-8	New Year's Card	32
Program 4-9	Birthday Greeting	33
Program 4-10	Birthday Cake Card	33
Program 4-11	Halloween Card	34
Program 4-12	Halloween Cat	34
Program 4-13	Mother's Day Card	35
Program 4-14	Easter Card	35
Program 4-15	Easter Bunny	36

Chapter 5**THE PERSONAL TOUCH****37**

Program 5-1	Short Story with Your Name	37
Program 5-2	Homework Assignment Notice	38
Program 5-3	Legal Notice	38
Program 5-4	Extra! Extra! Read All About It!	39
Program 5-5	Computer Interviewer	40
Program 5-6	Doctor's Assistant	41
Program 5-7	Tax Collector's Assistant	41
Program 5-8	Simple Arithmetic Quiz	42
Program 5-9	Arithmetic Quiz with Two Questions	43
Program 5-10	Geography Quiz	43
Program 5-11	Geography Quiz with an Easy Question	44

Chapter 6**DRAWING STATIONARY OBJECTS****45**

Program 6-1	Asterisks at Five Positions	46
Program 6-2	Stationary Eyes	48
Program 6-3	Stationary Eyes, Alternate Form	48
Program 6-4	Stationary Alien	49
Program 6-5	Stationary Bug	50
Program 6-6	Stationary Bug, Short Form	51

Chapter 7 MAKING PICTURES MOVE 52

Program 7-1	Asterisk Moving Right, without Erasure	52
Program 7-2	Asterisk Moving Right, with Erasure	52
Program 7-3	Asterisk Moving Right, Alternate Method	53
Program 7-4	Asterisk Moving Left, without Shadow	53
Program 7-5	Eyes Moving Right	53
Program 7-6	Alien Moving Right	54
Program 7-7	Alien Moving Right, with Spaces	55
Program 7-8	Car Moving Right	56

Chapter 8 SOME THINGS MOVING WHILE OTHERS REMAIN STATIONARY 57

Program 8-1	Small Rocket Moving Right	57
Program 8-2	Tank Firing a Rocket to the Right	58
Program 8-3	Arrow Shot at a Target	59
Program 8-4	Moving Ball	59
Program 8-5	Stick Man Throwing Ball	60
Program 8-6	Stick Man Playing Handball	60
Program 8-7	Space Station Firing a Small Missile	61
Program 8-8	Space Station Firing a Large Missile	62

Chapter 9 MAKING CREATURES WALK 63

Program 9-1	Stationary Bug with Moving Legs	64
Program 9-2	Bug Walking Right	65
Program 9-3	Bug Walking Left	66
Program 9-4	Walking Bug with Blinking Eyes and Open Mouth	66
Program 9-5	Funny, Moving Aliens	67
Program 9-6	Stationary Creature Making Faces	68

Chapter 10 A UNIVERSE OF CHANGE 70

Program 10-1	Snowfall	71
Program 10-2	Bugs Appearing and Disappearing at Random Positions	72
Program 10-3	Exploding Stars at Random Positions	73
Program 10-4	Random Characters at Random Positions	74
Program 10-5	Random Four-Letter-Word Generator	75

Chapter 11 **CONTROLLING ACTION FROM THE KEYBOARD** 76

Program 11-1	Space Station Firing a Missile, under Keyboard Control	77
Program 11-2	Eye Creatures Moving Right	78
Program 11-3	Target Practice	78

Chapter 12 **MAKING NOISE** 81

Program 12-1	Simple Sound	81
Program 12-2	Random Sounds	82
Program 12-3	Computer Song	82
Program 12-4	Rocket Launch with Sound	83
Program 12-5	Rocket Launch with Noise	83
Program 12-6	Eye Creatures Making Noises	84
Program 12-7	Blinking Eye Creatures Making Noise	84
Program 12-8	Eye Creatures Moving Slowly	85
Program 12-9	Moving Alien Making Sound	86

Chapter 13 **FALLING OBJECTS** 87

Program 13-1	Falling Asterisk without Erasure	88
Program 13-2	Falling Asterisk with Erasure	88
Program 13-3	Falling Spider	89
Program 13-4	Falling Alien	90
Program 13-5	Blinking, Falling Alien	91
Program 13-6	Falling Alien Blinking Madly	91
Program 13-7	Race Car Moving Down	93
Program 13-8	Race Car Moving Down, then Right	95
Program 13-9	Falling Spaceship	96
Program 13-10	Falling Spaceship in Random Columns	97
Program 13-11	Alien Spaceship Moving Up and Down in Random Columns	98

Chapter 14 **BOMBING GAME** 99

Program 14-1	Falling Bomb	99
Program 14-2	Bomb Falling to the Ground, with Noise	100
Program 14-3	Small Bomb Falling, with Noise	101
Program 14-4	Bombing Under Keyboard Control	102
Program 14-5	Moving a Bomb Before Dropping It	103
Program 14-6	Bombs Away Game	104
Program 14-7	Alien Attack Ship Dropping Bombs	106
Program 14-8	Alien Attack Ship Dropping Bombs, with Explosion	106

Chapter 15 DIAGONAL MOTION 108

Program 15-1	Eye Creature Moving Right Diagonally	108
Program 15-2	Ball Rolling Down Chute	108
Program 15-3	Alien Moving Diagonally, with Sound	110
Program 15-4	Asterisk Moving Diagonally from Column 5	111
Program 15-5	Pulsating Alien Moving Right Diagonally from Column 5	112
Program 15-6	Eye Creature Moving Left Diagonally	113
Program 15-7	Car Traveling Along a Highway	113
Program 15-8	Eye Creature Moving Diagonally, Right to Left, from Top	115
Program 15-9	Traffic Moving Along a Highway	115

Chapter 16 USING CUSTOM CHARACTERS 117

Program 16-1	Stick Figure Moving Right	123
Program 16-2	Rocket Moving Right	125
Program 16-3	Race Car Moving Down	127
Program 16-4	Pinchbug Moving Right	130
Program 16-5	Pinchbug with Moving Jaws, Traveling Right	136
Program 16-6	Stick Figure Dancing	137
Program 16-7	Stick Figure Dancing Across the Screen	138
Program 16-8	Torg and His Terrible Dance	140

Chapter 17 COMBINING PROGRAMS 142

Program 17-1	Moving the Master Spaceship	144
Program 17-2	Master Ship with Bomb Underneath	146
Program 17-3	Master Ship Attacking with Guided Bomb	147
Program 17-4	Spaceship Game with Speed Control	148
Program 17-5	Spaceship Game with Speed Control and Explosions	149

Chapter 18 CONTROLLING MOTION IN ALL DIRECTIONS 152

Program 18-1	Moving a Bomb	153
Program 18-2	Starship Moving Anywhere	160
Program 18-3	Starship Moving Anywhere, with Sound and Targets	162
Program 18-4	Starship Game with Momentum	164

Chapter 19 TARGET PRACTICE 166

Program 19-1	Shell Rising	171	
Program 19-2	Shell Falling	172	
Program 19-3	Shell Rising and Falling	172	
Program 19-4	Artillery Gun Shelling a Factory		173
Program 19-5	Artillery Gun Shelling a Factory, with Explosion	174	
Program 19-6	Jet Fighter Attacking	178	
Program 19-7	Jet Fighter Attacking Spaceship		179

Chapter 20 SHOOTING AT A MOVING TARGET 182

Program 20-1	Simple Target Moving Up and Down		182
Program 20-2	Tank Firing a Bullet	184	
Program 20-3	Tank Firing a Bullet at a Target	185	
Program 20-4	Target Practice with a Moving Target		186
Program 20-5	Target Practice with a Moving Alien Ship		186
Program 20-6	Target Practiced with a Laser Blast	187	

Chapter 21 MOVING TWO THINGS SIMULTANEOUSLY 189

Program 21-1	Ship and Submarine Moving in Different Directions	190	
Program 21-2	Ship, Submarine, Ocean Waves, and Birds		192
Program 21-3	Submarine Firing a Missile	194	
Program 21-4	Ocean, Birds, Ship, and Submarine Firing Missiles	194	

Chapter 22 RACE CAR GAME 197

Program 22-1	Beginner's Racetrack	197	
Program 22-2	Car Racing Around a Track		201
Program 22-3	Intermediate Racing Game		204
Program 22-4	Advanced Racetrack	206	

Chapter 23 STARSHIP GAME 207

Program 23-1	Stationary Starship	208	
Program 23-2	Starship Moving Along the Bottom		209
Program 23-3	Starship Firing a Laser Blast at Targets		211
Program 23-4	Starship Attacking Targets, Including Doomsday	213	

Chapter 24	PLANETARY ROVER GAME	214
Program 24-1	Planetary Rover Moving Right and Left	216
Program 24-2	Planetary Rover with Falling Meteorite	218
Program 24-3	Planetary Rover Game	220
Chapter 25	USING COLOR	221
Program 25-1	Kaleidoscope of Screen Colors	222
Program 25-2	Kaleidoscope of Colors with Characters	223
Program 25-3	Red-and-White Alien Moving Diagonally	224
Program 25-4	Solid Red Line Across a White Screen	225
Program 25-5	Red House with a Green Roof	227
Program 25-6	Little Red Schoolhouse	227
Program 25-7	Bouncing Ball Changing Color	228
Program 25-8	Random Multicolored Exploding Stars	234
Program 25-9	Random Art with Color	236
Appendix	TI COLOR SETS INDEX	237 243

EDITOR'S FOREWORD

What can you say about a book with program titles like these?

Birthday Cake Card
Tax Collector's Assistant
Stationary Bug
Playing Handball
Exploding Stars at Random Positions
Random Four-Letter-Word Generator
Falling Spider
Alien Attack Ship Dropping Bombs
Traffic Moving Along Highway
Pinchbug with Moving Jaws, Traveling Right
Master Ship Attacking with Guided Bombs
Artillery Gun Shelling a Factory
Starship Firing a Laser Blast at Targets
Bouncing Ball Changing Color

Tony Fabbri's programs speak for themselves. This book is light, enjoyable, easy to read, and probably addicting as well. For those serious souls who are worried about wasting valuable time, let me add that the book also teaches BASIC and good programming practices. What more can you ask? I am sure you will have as good a time with this book as I did. Please remember to tear yourself away from your TI-99/4A often enough to remain vaguely connected with the outside world.

Lance A. Leventhal
San Diego, CA

PREFACE

Although many books deal with computers, hardly any are just plain fun to read and apply. This book does not explain the insides of computers or describe all possible ways to use them. Rather, it lets you do exciting things with the computer and see action unfold on the screen. It assumes no prior knowledge of computers or programming. You will learn about computers and enjoy doing it.

After introducing a few fundamentals, the book slowly leads the beginner through drawing and moving pictures such as aliens, rockets, animals, insects, spacecraft, and creatures. Then we introduce sound, and the pictures come to life. Spaceships, airplanes, tanks, and strange creatures make noises as they move, turn, fire, blink, and attack. We then introduce the idea of controlling the action from the keyboard. From then on, you are able to create your own simple arcade games.

All the discussions and early programs are short. You will find it easy to use the examples to create new games. A significant fringe benefit of this fun is that you will actually be learning the skills needed for practical applications in business, education, engineering, management, and science.

This particular version of the book uses the popular TI-99/4A home computer. It assumes at least a 16K system. No special graphics equipment is necessary, since we draw all pictures with ordinary PRINT statements. You can simply enter a program and watch the action on the screen.

I hope this book provides many hours of enjoyment for you, both as a learning tool and as a reference. I hope it also encourages you to build a comfortable relationship with the computer and thus free your imagination for creative work. This is the key to obtaining the greatest benefit from today's amazing personal computers.

Tony Fabbri

ANIMATION, GAMES,
AND SOUND
FOR THE
TI 99/4A

INTRODUCTION

The Texas Instruments TI-99/4A home computer is a popular, inexpensive personal computer. Although it is only the size of a portable typewriter, it is as powerful as computers of the 1950s and 1960s that occupied entire rooms and cost hundreds of thousands of dollars. Together with a television set (preferably color) and a cassette recorder, it forms a complete system. Among its many uses are:

1. Word processing (electronic writing of letters, notices, reports, and books without erasures, misspellings, and typing errors).
2. Business and financial calculations, such as figuring interest rates, loan payments, rates of return, and cost of capital.
3. Computerized lessons in arithmetic, spelling, social studies, language, music, art, science, reading, and computer programming.
4. Space, adventure, sports, gambling, word, mathematical, business, and skill games.
5. Record keeping for collectors, hobbyists, investors, property owners, and officials in local clubs and organizations.
6. Preparing a home budget.
7. Analyzing and tracking investments such as stocks, bonds, real estate, and commodities. The TI-99/4A can even obtain current quotations and financial news over the telephone.
8. Balancing checkbooks, helping with tax forms, calculating percentages and discounts, and performing other home financial tasks.
9. Performing engineering calculations for determining thrust, inertia, stress, energy use, distances, and flow rates.

10. Performing business functions such as inventory management, cash management, payroll, and general ledger.
11. Helping teachers and school administrators with such tasks as recording student data, keeping attendance, grading, scoring tests, scheduling classes, measuring student progress, maintaining inventory records, and preparing budgets.
12. Controlling things such as lights, motors, generators, household appliances, and laboratory or shop equipment.
13. Creating art, music, and sound effects.
14. Electronic filing of names and addresses, receipts, and appointments.
15. Automatic generation of form letters, mailing labels, flyers, and circulars.

These are only a few of the TI-99/4A's many uses. Hundreds of thousands of people have purchased TIs for use in business, education, recreation, or the home.

COMPONENTS OF THE TI-99/4A

A standard TI-99/4A (see Figure 1-1 for a photograph) consists of two pieces of equipment (we call this the *hardware*):

1. The computer itself, a slim rectangular black and silver plastic box with a sloping front. On the left side of the sloping front are black keys arranged as on a small portable typewriter. On the right side is a slot for special program cartridges. On the front edge at the far right are a small power indicator and an on/off slide switch.
2. A television set that serves as an output device or display unit. We call this the *video display*.



Figure 1-1 The TI-99/4A Computer System

You can buy many accessories for the TI-99/4A; among the most popular are:

1. **Cassette recorder.** This lets you play cassettes into the computer and record them from it. You may use almost any recorder, but it must have a counter.
2. **Disk drive.** This is a small box with a rectangular slit in the front. It serves the same purpose as a cassette recorder, but instead of cassettes it uses thin, flexible media called *diskettes*. A diskette is about the size of a 45-rpm record. A disk drive is more expensive than a cassette recorder, but it is faster, more reliable, and more convenient to use.
3. **More memory.** This is always useful, because it lets the computer remember more things. You may compare additional memory to extra file drawers or cabinets.
4. **Printer.** Printers are relatively expensive, but they give you output on paper (*hard copy*) that you can save for later use, put in an envelope, or hand in as a complete school assignment.
5. **An expansion box.** This is a large silver box that expands the computer's capabilities, allowing you to add more memory, disk drives, and a printer.
6. **A voice synthesizer unit.** This unit, which plugs directly into the computer, provides a surprisingly natural-sounding voice. The computer can then talk to you just as in science fiction movies.

COMPUTER

First, let us discuss the computer. The power connection is in the back. You should connect the TI-99/4A to a wall socket using the power supply unit. On the front edge of the computer at the far right are the on/off slide switch and a power indicator light. You can turn the computer off by sliding the on/off switch left, disconnecting the power supply, or pulling the power supply plug out of the wall socket. You shouldn't do any of these often, but pulling the plug out of the wall socket is the surest and most reliable approach. It also turns the power supply off. Unfortunately, the computer's power light is so dim in normal room lighting that you usually cannot tell if it is on or off.

The keys on the front allow you to enter things into the computer. The only fundamental difference from a typewriter is that the entries appear on the screen rather than on paper.

That is all you can see of the computer. The electrical parts are inside the case and are organized much like the nervous system of a person. These parts consist of:

1. A brain, generally called the *central processing unit*, or *CPU*.
2. **Memory.** Its only special feature is that it forgets everything when you turn the computer off. That's why you don't want to turn the power off very often. When you quit for the day, you should save anything you might need on a cassette or diskette.
3. **Connections to the outside world, generally called *interfaces*.** These are like the body's nerves and muscles.

TELEVISION DISPLAY

The television set serves as the computer's output device. The screen can hold 23 lines of 28 characters (a *character* is any letter, number, or symbol). The lines are thus quite short, but the individual characters are large and easy to read. The TI-99/4A can also draw pictures (or *graphics*, to use the computer term). For graphics, the screen holds 24 lines of 32 characters; we can think of it as a grid (see Figure 1-2) consisting of 24 lines or rows and 32 columns. We will refer to the rows as 1 through 24, starting with 1 at the top, and to the columns as 1 through 32, starting with 1 at the left. You will find it handy later to have copies of the grid in Figure 1-2, since we will use it to draw pictures and position the characters.

On the typewriter, the typing mechanism or carriage indicates where your next entry will appear. To provide a similar indicator on the screen, the computer must create a moving marker. We call this marker the *cursor*. The TI's cursor is a blinking square that is always one character space beyond whatever you last typed.

Having results appear on a screen rather than on paper may seem strange at first. You will quickly learn to like the ability to change or erase the screen easily. Changes and

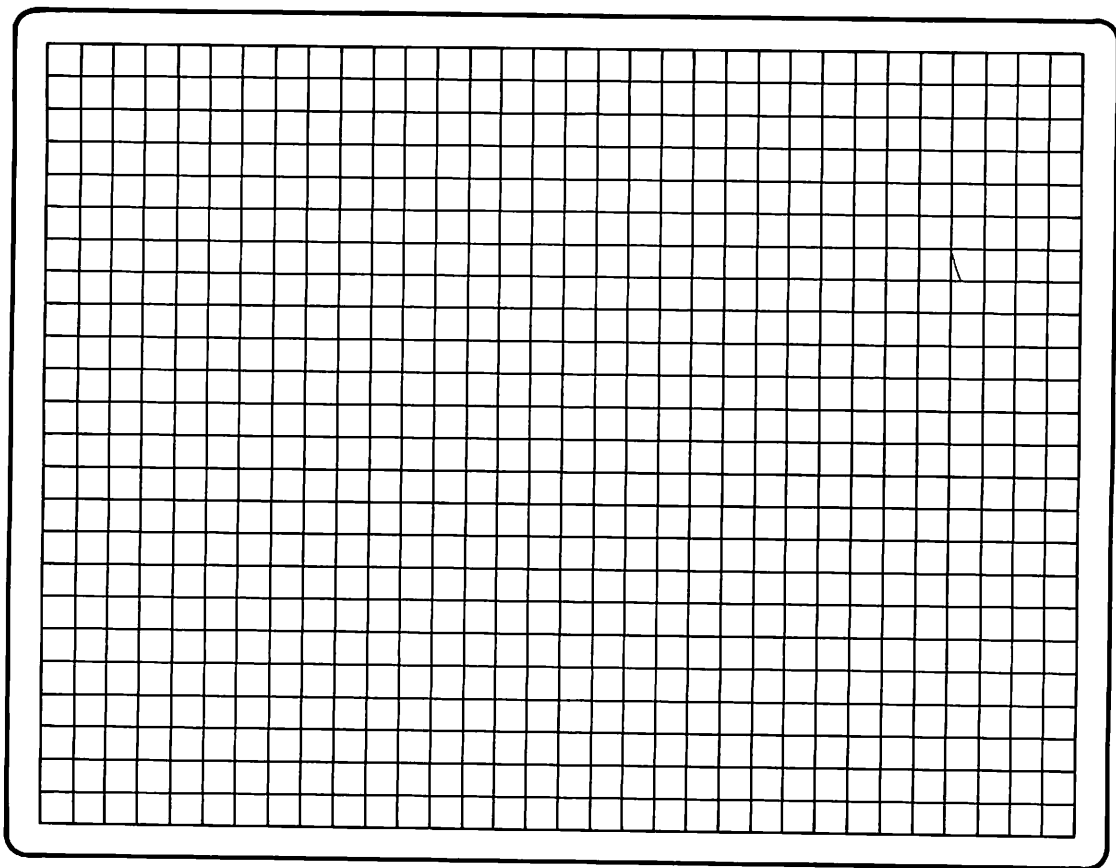


Figure 1-2 Grid of Screen Locations for Pictures on the TI-99/4A

corrections are faster, neater, and simpler on a screen than on paper. In fact, if you work with a screen for a while, going back to making changes or corrections on paper will seem like carving letters in a stone tablet. But there is a drawback—the screen does not provide you with a permanent record of your work. What if you want to see something you did an hour ago or yesterday? Furthermore, you cannot send someone your television set in an envelope (particularly not while it's still plugged in and connected to the computer). So it is always nice to have a printer, even though we won't spend much time discussing it.

One special feature of the TI's display is an automatic "screen saver" facility. If you do not press a key for a while, the computer blanks the screen to reduce wear and tear on the television set. So, if you leave the computer for any length of time, you will probably return to a blank screen. Don't worry. You can restore what was there by pressing any key (the space bar is usually a safe bet).

KEYBOARD

The keyboard (see Figure 1-3) resembles a small portable typewriter. As on a typewriter, you can use the SHIFT keys (one on each side in the row above the space bar) to type capital letters and uppercase symbols. Note that the right-hand SHIFT key is much larger than the left-hand SHIFT key. You can also press ALPHA LOCK down (locking it in place) to produce capital letters without pressing SHIFT each time. However, you can see right away that this is not a normal typewriter keyboard. Among its special features are:

1. Extra keys such as CTRL (just to the right of the space bar) and FCTN (left of the space bar).



Figure 1-3 TI-99/4A Keyboard

2. Markings such as arrows, brackets, dashes, apostrophes, and other symbols on the front of some keys. These compensate for the shortening of the keyboard to make room for the cartridge outlet.
3. Colored dots on the front of the CTRL, FCTN, and ENTER keys. The gray dot on the front of the FCTN key corresponds to the gray dot on the plastic strip you insert in the track above the number keys. This strip reads DEL, INS, ERASE, CLEAR, BEGIN, PROC'D, AID, REDO, BACK, and QUIT, from left to right.

What does all this mean? Well, let us describe it little by little. We will not tell you everything at once—just the high points. You can learn the rest later as you do things.

Type a Little Something

To see how the TI-99/4A works, turn it on by sliding the power switch to the right. You will see a colored display with the Texas Instruments logo and the message READY-PRESS ANY KEY TO BEGIN. Press the space bar. After a brief delay, you will see the logo again and the message

PRESS

1 FOR TI BASIC

Press 1. After another delay, you will see the words TI BASIC READY and, on the next line, > and a flashing black square (the cursor). All this will be in a lower left-hand corner of your screen. Now enter the typist's standard phrase:

Now is the time for all good men to come to the aid of their country.

Press SHIFT and N to produce a capital letter, just as you would on a typewriter. Be careful when you reach for the left-hand SHIFT key; it is not quite where a typist would expect to find it. You could easily press CTRL or ALPHA LOCK instead; note that ALPHA LOCK is below SHIFT, not above it as SHIFT LOCK usually is on a typewriter. What happened to the lowercase letters? They do not appear in their usual form; instead, they look like miniature capital letters.

Keep right on typing. Don't worry about mistakes, and don't stop when you reach the end of the line. The computer will proceed to the next line automatically. When you finish, press the ENTER key (just above the right-hand SHIFT key—it has a yellow dot on its front). The computer will respond with

* INCORRECT STATEMENT

Don't worry about this right now. It just means the computer does not recognize typing exercises or statements involving complex human feelings such as patriotism. In fact, it understands only rather simple commands.

Now try typing the following sentence:

The first (grand) prize is \$1,500.

This will give you practice with the uppercase symbols (,), and \$, as well as with some punctuation marks. Press ENTER.

Now press ALPHA LOCK (lower left-hand corner) until it catches. Type the last sentence again. Note that the letters are all capitals now, but you must still press SHIFT to type (,), and \$. ALPHA LOCK locks in capital letters but does not affect nonletter keys; it works differently from SHIFT LOCK on a typewriter. Press ENTER and ignore the computer's standard INCORRECT STATEMENT remark.

Now try entering the following line (including all the punctuation):

He said, "Won't you come and play?"

This is fine until you reach the quotation mark. Where is that symbol? It is not in any of its common positions on a typewriter. Instead, it is on the front of the P key. To enter it, press FCTN (lower right-hand corner) and P simultaneously. This is somewhat like entering a shifted character, but you will probably find it quite awkward if you are a typist. You may feel as though you have to cross your left hand over your right to reach FCTN. A little practice will improve the situation, but this strange procedure will stop you from doing much touch-typing on the TI-99/4A.

You need the FCTN key to enter other common symbols as well. For example, you enter ' by pressing FCTN and O simultaneously, and ? by pressing FCTN and I simultaneously. Table 1-1 contains a list of the symbols that require the FCTN key.

**TABLE 1-1 DISPLAYING CHARACTERS
USING FCTN**

To Get This Symbol	Press FCTN and
-	W
[	R
]	T
—	U
?	I
'	O
"	P
:	A
{	F
}	G
\	Z
`	C

Note the following other unusual features of the computer's keyboard:

1. All keys repeat. If you press a key and release it, you will get that character entered once. If, however, you hold the key down for a while (longer than the one second advised in your TI manual), the key will begin to repeat, and you will see a string of characters across the screen.
2. The zero on the zero key has a slash through it, but the 0 appears on the screen without the slash. It does have rounded edges, while O (capital letter) has square edges.

3. Since lowercase letters appear as small capitals, you cannot confuse the number one (1) with lowercase L (L).
4. The symbol you enter by pressing FCTN and U is an underscore (_), even though it looks like a hyphen or dash on the front of the key. The hyphen or dash is the uppercase symbol on the key just right of the P key.
5. The symbol on the front of the C key is an opening single quotation mark or reversed apostrophe. The apostrophe is on the front of the O key.
6. The symbol on the front of the A key is a vertical line with a break in the middle. The exclamation point, on the other hand, is the uppercase symbol on the I key.
7. The slash symbol / (a diagonal line from lower left to top right) is on the key just right of the P key. The symbol on the front of the Z key is a "reversed slash" (\), a diagonal line from top left to lower right.

TALKING TO A COMPUTER

Now that we have described the computer a little, we are ready to tell it to do something. But how do you talk to a computer? Does it understand French, Spanish, or Swahili? The answer is none of these. Computers have their own languages, with names like BASIC, COBOL, FORTRAN, and Pascal. The one we will use, called BASIC, is the most popular language for small computers. We will just briefly introduce BASIC here and tell you more about it as we go along, rather than describing it completely right away. After all, no one tells you all about English in the first grade.

A First Look at BASIC

BASIC lets us enter instructions, or *statements*, into the computer's memory. A set of instructions (a *program*) forms a procedure that the computer can follow. You can compare programs to recipes or kit assembly instructions.

The essential features of BASIC are:

1. Each line (or *statement*) must start with a number. We usually choose these line numbers at least 10 apart (for example, 100, 110, 120, 130) to permit later insertions (such as 105 or 136). You end a line (as far as the computer is concerned) by pressing the ENTER key. This puts the line in the computer's memory. ENTER acts like the carriage return on a typewriter, although there is, of course, no physical carriage to move.
2. The computer does what the lines say in numerical order. That is, it starts with the lowest-numbered line and proceeds upward, unless specifically told to do otherwise. You can enter the lines in any order; the computer arranges them in numerical order automatically.

Besides statements, BASIC also has commands that tell the computer what to do with the instructions; these do not have numbers. The most common commands are:

NEW clear the computer (start with a clean slate).

LIST Display the current program, place the lines in numerical order.

RUN Do what the program says, starting with the lowest-numbered statements and continuing in numerical order.

So, to enter a BASIC program and have the computer do what you say, you must:

1. Type **NEW** to clear the computer. Be sure you really want to do this, since any work you have in the computer will be lost.
2. Enter the statements with their line numbers in any order.
3. Type **RUN** to have the computer execute the program.

You may type **LIST** at any time to get an ordered listing of the program. Remember to end each line, regardless of whether it is a command or a statement, by pressing the **ENTER** key.

CORRECTING TYPING ERRORS

So far, we have not concerned ourselves with typing errors. Fortunately, the TI-99/4A allows you to correct these errors easily; it is much better in this regard than an expensive correcting typewriter.

When you see an error in something you just typed, the first thing you must do is move the cursor back to it. You can move the cursor (without changing any characters) left with the left arrow and (believe it or not) right with the right arrow. The left arrow is on the front of the **S** key, so you obtain it by pressing **FCTN** and **S** simultaneously. You obtain the right arrow by pressing **FCTN** and **D** simultaneously. Try both of these, but remember that keys repeat if you hold them down too long.

When the cursor reaches the error, simply type the correct symbol if all you need is a simple replacement. If, for example, you typed **MOW** instead of **NOW**, move the cursor to the **M** and press **N**. The new letter replaces the old one immediately; you do not have to erase the old one first as you would on a correcting typewriter.

But what if you typed an extra letter, say **FOUR** instead of **FOR**? To erase (or *delete*) a letter, press **FCTN** and **1** (note the **DEL** on the plastic strip above the **1** key). The letter vanishes from under the cursor, and everything right of it moves left to fill the gap.

If, on the other hand, you must add a letter (say you typed **BAD** instead of **BEAD**), you must press **FCTN** and **2** (note the **INS**, for *insert*, on the plastic strip above the **2** key). Nothing happens on the screen. However, when you now press a character key, the character appears where the cursor was, while the cursor and the subsequent text move right to make room for it. Remember, if you do not press **INS** (**FCTN** and **2**), you will overwrite the character you had, rather than add a character. In fact, you can insert as many characters as you want; to stop inserting, either press **ENTER** or move the cursor with the arrow key.

One problem with inserting is that you may forget whether it is active. The screen does not indicate whether the next character you type will be inserted or will overwrite an old character. All you can do is type, and correct the error if you do not get what you expected.

If a line contains many errors (say you had your fingers placed wrong), you might as well start over. To erase the current line, press FCTN and 3. Note the word ERASE on the plastic strip above the 3 key.

What if you press ENTER before you see the error (or maybe the computer complains about some line you entered long ago)? One alternative is to type the line again with the same line number. The computer replaces the old line with the new one automatically. Note that if you want simply to erase the line completely, enter its number and press ENTER. Another alternative is to type the word EDIT (or `EDIT`), followed by the line number and ENTER. The line will appear at the bottom of the screen, and you can move the cursor and correct it.

What if you don't see the mistake at all, and you press ENTER or type RUN? No, the computer will not emit black smoke or start to cry. It will just say something like INCORRECT STATEMENT, BAD NAME, or CAN'T DO THAT. This means you typed something it did not understand (such as PRUNT instead of PRINT or RUM instead of RUN) or could not do. In most simple cases, the mistake is obvious and you can reenter the line and start over.

QUITTING TIME

To exit from TI BASIC, press FCTN and the = key (upper right-hand corner). The computer will return to its original multicolored screen with the TI logo. Be careful about doing this; it erases any program you had in memory. Also be cautious when you want to enter a + (the uppercase symbol on the = key); if you press FCTN instead of the right-hand SHIFT key, you will lose your program. An alternate way to exit is to type B, Y, E, and press ENTER.

HELPFUL HINTS

Until you get used to it, the TI keyboard is confusing. It has extra keys, and some keys are not in the same locations as on a typewriter. Furthermore, you must remember the meaning of the FCTN and SHIFT keys and the uses of the plastic strip at the top. Don't worry; it may be enough to make your head swim now, but you will quickly become familiar with it. And you will soon be amazed at what you can do; it just takes a little time.

As with most tasks, there are a few things you should remember. We therefore conclude this chapter with a list of those; refer to it whenever you are not quite sure what to do next. In a short time, these things will become second nature.

1. To stop the computer when you are not sure what it is doing, press the FCTN and 4 (CLEAR) keys simultaneously.
2. To stop the computer and start over from scratch, turn it off and then back on again. Do this only as a last resort, and never while playing a disk or cassette. You can also press the FCTN and = (QUIT) keys simultaneously instead.
3. To type an uppercase symbol or a capital letter, press the key while holding a SHIFT key down.

4. To print a character on the front of a key, press FCTN and the key.
5. To enter lowercase letters, release the ALPHA LOCK key (that is, let it go to its normal up position). To enter capital letters without pressing SHIFT, push ALPHA LOCK down until it engages in the down position. Remember that ALPHA LOCK does not affect nonletter keys.
6. To correct a typing error, move the cursor to it using the arrow keys (FCTN and S is left, FCTN and D is right) and retype everything. If you have already entered the line into the computer, just retype it (or enter EDIT and the line number to get it back).
7. To display your current program in numerical order, enter LIST.
8. To clear out old programs and start over, enter NEW. This also clears the screen.

2

SIMPLE PRINTING

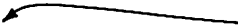
Before starting sessions with the computer, be sure ALPHA LOCK is engaged (down). This not only makes the letters larger and easier to see, but it is also essential to avoid errors in some situations. The computer is occasionally fussy about lowercase letters.

PRINT STATEMENT

PRINT tells the computer something on the screen. The sequence

```
100 PRINT "DOG"  
RUN
```

Press FCTN and P to type."

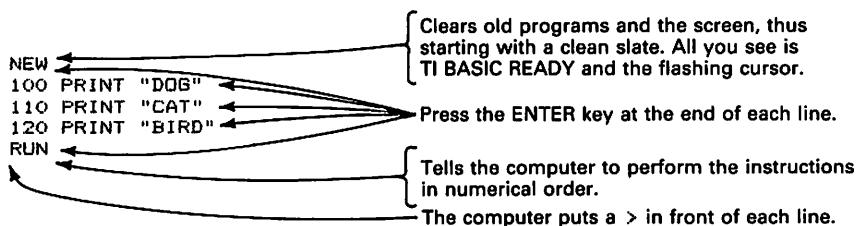


prints the word DOG. We can put anything between the quotation marks. For example,

```
100 PRINT "I AM A CAMERA"  
RUN
```

puts the message I AM A CAMERA on the screen. To see how PRINT works, enter Program 2-1.

Program 2-1 Print Animal Names



When you enter RUN, the screen turns light green (apparently the computer has trouble digesting programs). The words

```
DOG  
CAT  
BIRD
```

appear. The screen then returns to its usual blue, and the statement

```
** DONE **
```

appears, indicating that the program had no errors. If an error occurs, the computer tells you the problem (for example, BAD NAME, CAN'T DO THAT, or INCORRECT STATEMENT) and where it was. If this happens, type LIST, correct the error (usually a typing mistake), then enter RUN again.

Note that the computer prints the results of Program 2-1 starting at the bottom of the screen. Run the program several times. Note that the old results move up the screen (a process called *scrolling*) as new results appear at the bottom.

Whatever you put inside quotation marks in a PRINT statement appears on the screen. To print several things on a line, separate them with commas or semicolons. Commas space things out, and semicolons draw them together. For example, type

```
NEW  
100 PRINT "DOG", "CAT"  
RUN
```

The result should look like this:

```
DOG      CAT
```

Note how far apart the words are. To bring them together, replace the comma with a semicolon. Type EDIT 100 and press ENTER to retrieve the line. Then move the cursor to the comma, type a semicolon over it, and press ENTER. Note that the cursor starts at the P in PRINT, not at the left margin.

```

NEW
100 PRINT "DOG"; "CAT"
RUN

```

Place a semicolon here.

The screen should look like this:

```

DOGCAT

```

The words are crowded together. Of course, you could separate them by putting spaces inside the quotation marks.

A comma makes the computer move to the next printing area. The screen consists of two areas, each of which holds 14 characters.



Semicolons make the computer crowd things together with no spaces in between. Now try

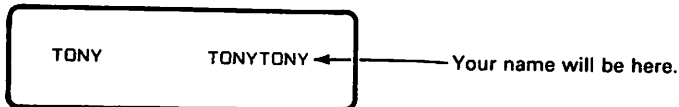
```

NEW
100 PRINT "TONY"; "TONY"; "TONY"
RUN

```

Put your name here.
Be sure to type a quotation mark before and after your name.

The screen should look like this:



This description assumes you have a short name like TONY, DICK, or HARRY (or SUE, JANE, or CAROL). If, on the other hand, your name is RUMPLESTILTSKIN, TUTANKHAMEN, or ANASTASIA, the computer will put it on the next line if it will not fit in the available space.

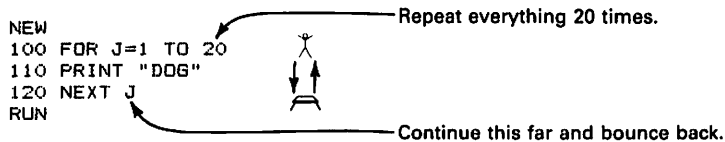
So far, we have only asked the computer to print one line. If we want it to print many lines, we must either enter the instructions one after another (if the lines are different) or tell it to repeat instructions (if the lines are the same). The FOR-NEXT statement tells the computer to repeat instructions. Obviously, we need to tell the computer what to repeat and how many times to repeat it. Therefore, FOR-NEXT has two parts:

1. FOR tells the computer how many times to repeat instructions. The usual form is FOR J=1 TO N, where N is the number of repetitions. We call J the *loop variable* or *index*.

2. NEXT (further along in the program) tells the computer how far (from FOR) to repeat. The usual form is NEXT J, where J is the loop variable in FOR.

For example, look at the following program. Compare the computer's action with that of a person bouncing on a trampoline.

Program 2-2 Print DOG 20 Times



Program 2-2 tells the computer to do everything between FOR J (line 100) and NEXT J (line 120) 20 times. The result is that the computer prints DOG 20 times. Note that the screen stays green as long as the program is running.

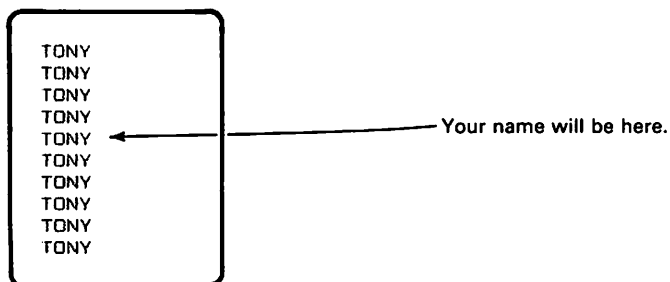
NEXT acts like a trampoline that bounces the computer back up to FOR. Every time it reaches NEXT J, it bounces back up to FOR J=1 TO 20. Programmers call this moving down and bouncing back a *loop*. After the 20th repetition, the computer falls through NEXT J to the line below. Now try printing your name ten times.

Program 2-3 Print Your Name Ten Times

```
NEW
100 FOR J=1 TO 10
110 PRINT "TONY"
120 NEXT J
RUN
```

Put your name here.

The screen should look like this:



To print your name 15 or 20 times, simply change the number after TO in line 100 to 15 or 20.

Put 500 after TO in line 100 and run the program again. Now the computer will keep printing your name for almost a minute. After it prints 23 lines, it will scroll the

To interrupt the program, press FCTN and 4 simultaneously; the plastic strip above the number keys indicates that this combination means CLEAR. If the computer keeps going, press FCTN and 4 again.

Program 2-4 Your Name in Columns

The diagram shows a BASIC program with four lines of code. Arrows point from handwritten annotations to specific parts of the code:

- An arrow points from the text "Tells the computer to do everything up to NEXT J 40 times" to the "NEXT J 40" part of line 100.
- An arrow points from the text "Be sure to type a comma at the end." to the comma after "TONY" in line 110.
- An arrow points from the text "Put your name here." to the space before the "NEXT J" in line 120.

```
NEW
100 FOR J=1 TO 40
110 PRINT "TONY",
120 NEXT J
RUN
```

The screen should show your name printed 40 times, twice on each line. Remember that the computer divides the screen into two areas, and a comma sends it to the next area.

[illegible]

Now try a semicolon.

Program 2-5 Your Name Filling the Screen

```
NEW
100 FOR J=1 TO 35
110 PRINT "TONY";
120 NEXT J
RUN
```

← Type a semicolon at the end.

← Put your name here.

The screen should show your name printed 35 times like this:

```
TONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONY
```

To fill the entire screen, change line 100 to

```
100 FOR J=1 TO 300
```

After a while, the screen will fill, and all you will see is the bottom line flashing. If your name does not fit evenly in 28 columns, the computer will print it as many times as it can and proceed to the next line. It will not truncate your name at the right.

To create a different effect, type * or – before and after your name. For example:

Program 2-6 Your Name Fancy

```
NEW
100 FOR J=1 TO 50
110 PRINT "*TONY*";
120 NEXT J
RUN
```



* is the shifted 8 key.

What do you see?

SAVING YOUR WORK

To save a program on a cassette, rewind the tape, set the location counter on the record to 000, move the tape to an unused area, and then type

```
SAVE CS1
```

The computer now controls the recording process. Follow the directions on the screen; you will see the following messages:

*REWIND CASSETTE TAPE THEN PRESS ENTER	← You have already done this when you moved the tape to an unused area. Simply press ENTER.
* PRESS CASSETTE RECORD THEN PRESS ENTER	
*RECORDING	← Wait for the computer to record the program.
*PRESS CASSETTE STOP THEN PRESS ENTER	
*CHECK TAPE (Y OR N)	← Answer this question by pressing Y (must be a capital letter). The computer will provide additional instructions for rewinding and checking the tape.

Be sure to keep the following information in a notebook for each program:

1. Beginning value of location counter (000 through 999).
2. Ending value of location counter (000 through 999).
3. Name.

To identify our programs, we have chosen the standard name format PROGRAM CC-NN, in which CC is the chapter number and NN the program number. You can pick whatever names you want because their only purpose is to help you identify programs. The computer does not care about names, but you should settle on a standard to avoid confusion.

The computer does not save the program names anywhere, so it is up to you to keep track of them. The only way you can tell your programs apart is by their tape counter locations and the names you give them. Do not store too many programs on a cassette because it takes time to wind and rewind the tapes.

To make it easier to identify a program, we will introduce a minor BASIC statement called REM (*Remark*). REM lines just explain to the computer user what the program is doing; the computer itself ignores them. In general, you may omit them to save typing, but you should put a REM at the beginning of each program to hold its name. This REM will always be line 100 in our programs so that we can determine which programs we have loaded by entering

LIST 100

LOADING A PROGRAM FROM TAPE

To load a program from a cassette, proceed as follows

1. Rewind the tape.
2. Position it just before the start of the program (use the tape counter to do this).
3. Type

OLD CS1

Note that this is really OLD to bring back an "old" program; it is not a misspelled LOAD.

4. Follow the directions on the screen. These are the same messages we saw when saving a program.

TAB STATEMENT

To form patterns on the screen, we need a way to tell the computer where to start printing. The TAB statement (line the tab on a typewriter) sets the starting position (column number) on a line.

For example, TAB(10) tells the computer to start printing in column 10, whereas TAB(J) tells it to start in column J. TAB thus lets the computer start anywhere on a line. Program 2-7 shows how we can use TAB to form a simple pattern.

Program 2-7 Your Name in a Diagonal Pattern

```
100 REM PROGRAM 2-7
110 FOR J=1 TO 10
120 PRINT TAB(J); "TONY"
130 NEXT J
RUN
```

Dash or hyphen is the uppercase character on the key just right of P.

Print 10 times.

Put your name here.

Starts in column 1 when J is 1.
Starts in column 2 when J is 2.
Starts in column 3 when J is 3.
Etc.

The screen should look like this:

```
TONY
 TONY
  TONY
   TONY
    TONY
     TONY
      TONY
       TONY
        TONY
         TONY
```

Your name will be here.

Each line starts one column farther right, since the J in TAB(J) is one larger. Note that the computer does not print in the column where the >'s appear.

To start the first line in column 10 and make subsequent lines move left, we use

```
FOR J=10 TO 1 STEP -1
```

J starts at 10 and is reduced by 1 each time the computer reaches NEXT J. This continues until reducing J makes it less than 1. Note that we must add the word STEP and the step size if the size is not 1.

Program 2-8 Friend's Name in a Diagonal Pattern

```

NEW
100 REM PROGRAM 2-8
110 FOR J=10 TO 1 STEP -1
120 PRINT TAB(J); "MARY"
130 NEXT J
RUN

```

Counts down (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

Put your friend's name here.

The screen shows

```

Column 10  MARY
Column 9   MARY
Column 8   MARY
Column 7   MARY
           MARY
           MARY
           MARY
           MARY
           MARY
           MARY
Column 1   MARY

```

The name moves left instead of right. We can combine Programs 2-7 and 2-8 to print two names in a crossing pattern.

Program 2-9 Two Names in a Crossing Pattern

```

NEW
100 REM PROGRAM 2-9
110 FOR J=1 TO 20
120 PRINT TAB(J); "TONY"
130 PRINT TAB(21-J); "MARY"
140 NEXT J
RUN

```

Put your name here.

Put your friend's name here.

[illegible]

21

3

ROCKETS FLY, BLAST OFF, AND CHASE ONE ANOTHER

Now let us use typed characters to produce simple drawings. For example, we can use slashes, hyphens, Vs, and the broken vertical line on the front of the A key to draw a rocket. The only new statement we need is one that clears the screen initially; this is `CALL CLEAR`.

Program 3-1 Rocket

```
NEW
100 REM PROGRAM 3-1
110 CALL CLEAR
120 PRINT " /\ "
130 PRINT " :| "
140 PRINT " :| "
150 PRINT " /  \"
160 PRINT " :| :| "
170 PRINT " :| :| "
180 PRINT " :| :| "
190 PRINT "-----"
200 PRINT " vv "
RUN
```

← Clears the screen

← / is just right of P, while \ is on the front of the Z key. Press FCTN and Z to type \.

← : is on the front of the A key. Press FCTN and A to type it.

← Use the letter V to form the engines.

The rocket appears to move up the screen as the computer draws it because printing is always done at the bottom. By having the computer print blank lines under it, the rocket appears to fly.

Program 3-2 Rocket Flying Fast

Do not type NEW !

```
100 REM PROGRAM 3-2
210 FOR J=1 TO 23
220 PRINT
230 NEXT J
240 CALL CLEAR
RUN
```

Makes rocket fly by printing blank lines under it.

Prints a blank line

Makes the computer erase the screen 1 last time before exiting.

To make smoke trail from the rocket's engine, change line 220 to

```
220 PRINT "  **  "
```

The asterisks under the engine represent smoke.

The rocket flies rather quickly. We can slow it down by making the computer pause before printing a blank line. The easiest way to do this is to force the computer into a tight loop in which nothing happens. The statements

```
FOR K=1 TO 200
NEXT K
```

form a tight loop. It works like telling a person to count from 1 to 200 before opening his or her eyes. Insert the lines

```
225 FOR K=1 TO 200
226 NEXT K
```

into Program 3-2 and run it. The rocket will move quickly while the computer is drawing it initially. It will then settle into a jerky slow motion. To speed it up, change the 200 in line 225 to 50. Experiment with this number until the motion looks reasonable (10 to 50 is a good range).

As you become more experienced in entering programs, you will probably start typing faster. Be careful that you do not enter lines faster than the computer can accept them.

After you press ENTER, wait for > and the cursor to appear. The computer will simply ignore anything you type before that occurs.

We can make figures chase each other up the screen by launching them one after another. Let's draw two friendly rockets being chased by an enemy rocket.

Program 3-3 Enemy Rocket Chasing Two Rockets, with Smoke

```
NEW
100 REM PROGRAM 3-3
110 CALL CLEAR
115 REM DRAW TWO FRIENDLY ROCKETS
120 PRINT "  /\  /\  "
130 PRINT "  ||  ||  "
140 PRINT "  ||  ||  "
150 PRINT "  ||  ||  "
160 PRINT "  --  --  "
170 PRINT "  vv  vv  "
175 REM FLY FRIENDLY ROCKETS WITH TRAILING SMOKE
180 FOR J=1 TO 23
190 PRINT " ** ** "
195 REM SLOW THE COMPUTER DOWN
200 FOR K=1 TO 25
210 NEXT K
220 NEXT J
225 REM DRAW THE ENEMY ROCKET
230 PRINT "  ^  "
240 PRINT "  / \  "
250 PRINT " !E! "
260 PRINT " !N! "
270 PRINT " !E! "
280 PRINT " !M! "
290 PRINT " !Y! "
300 PRINT "  --- "
310 PRINT "  V  "
315 REM FLY ENEMY ROCKET WITH TRAILING SMOKE
320 FOR J=1 TO 23
330 PRINT "  *  "
335 REM SLOW THE COMPUTER DOWN
340 FOR K=1 TO 30
350 NEXT K
360 NEXT J
370 CALL CLEAR
RUN
```

Annotations:

- Line 120: " /\ /\ " → 2 spaces
- Line 170: " vv vv " → The letter V represents the engines.
- Line 230: " ^ " → ^ is shifted 6.

One way to save on typing is to use the computer's automatic line numbering. You activate this feature by typing

NUM INITIAL, INCREMENT

where INITIAL is the first line number you want and INCREMENT is the gap you want between line numbers. For instance, to start with line 200 and space subsequent lines by 50 (that is, 200, 250, 300, and so on), you would enter

NUM 200,50

The computer is then in *Number Mode* and will generate line numbers automatically. This is handy by itself, but there is an even better shortcut. If you simply press ENTER after typing NUM, the computer starts with line 100 and an increment of 10. This default is exactly what we use normally anyway.

From here on, we will list all programs so that you can enter them using NUM. You should omit our comments; we have numbered them so that they can be skipped without

affecting the sequence. Just enter NUM and start with the program name on line 100. After you enter a line, the next line number will appear automatically. To stop automatic numbering (so you can LIST or RUN the program), simply press ENTER immediately after a line number appears.

Although automatic line numbering is convenient, it does lead to some new problems. Note the following:

1. Remember not to type a line number yourself. You will just have to move the cursor back and type over it. This is a surprisingly common mistake.
2. If you see an error after pressing ENTER, you must stop automatic line numbering to correct it. For example, if you see an error in line 240 after entering it, you must press ENTER to stop automatic line numbering, EDIT 240 to bring back line 240, and then NUM 250 to restart automatic numbering after making the correction.

Note that you can edit a line in Number Mode using the arrow keys, DEL, INS, and even ERASE, but you cannot change the line number. Pressing CLEAR (FCTN and 4) scrolls the current line up on the screen without entering it into the computer's memory; it also ends automatic numbering.

We can fly an alien spacecraft as follows:

Program 3-4 Alien Spacecraft Flying, with Smoke

```

NEW
100 REM PROGRAM 3-4
110 CALL CLEAR
120 PRINT "X-----X"
130 PRINT "I   @@@@ I"
140 PRINT "I   ==== I"
150 PRINT "X-----X"
160 PRINT "  ||  ||  "
170 PRINT "  --  --  "
180 FOR J=1 TO 23
190 PRINT "  **  **  "
200 FOR K=1 TO 25
210 NEXT K
220 NEXT J
230 CALL CLEAR
RUN

```

6 dashes

Of course no launch is realistic without a countdown to blastoff. To produce a countdown, add the lines

```

101 CALL CLEAR
102 FOR J=20 TO 1 STEP -1
103 CALL CLEAR
104 PRINT J; " <=== ";
105 NEXT J

```

< is the shifted , (comma) key.

Put a semicolon here.

To add these lines, first enter

```
NUM 101,1
```

then just type the statements; the computer will provide the line numbers automatically. When line 106 appears, press FCTN and 4 (CLEAR) to exit.

To slow the countdown, change the ending to

```
105 FOR K=1 TO 25
106 NEXT K
107 NEXT J
```

As the programs become longer, you will not have enough room on your screen for complete listings. To obtain a partial listing, type

```
LIST LOWER-UPPER
```

where LOWER and UPPER are line numbers in your program. You can omit LOWER or UPPER if you want to list everything from the beginning or through the end. For example, all the following commands work:

```
LIST 210-240 (lists lines starting with 210 and ending with 240)
LIST 210-    (lists line 210 and everything after it)
LIST -240    (lists everything up to and including line 240)
LIST 210     (lists only line 210)
LIST        (lists the entire program)
```

To draw a complex warship, use the picture

```
  ( )
   ( )
    ( )
  ( ) -- ( ) -- ( )
 ( ) \MARS/ ( )
    \00/
```

Other exercises you could try include:

1. A rocket chasing an alien spacecraft.
2. A squadron of rockets flying in formation.
3. An airplane flying off the screen.
4. A three-stage rocket.

4

ENTERING NAMES AND DRAWING GREETING CARDS

INPUT STATEMENT

INPUT statements stop the computer and let you enter letters or numbers. A typical example is

```
100 INPUT N$
```

The computer prints a question mark, then flashes the cursor, waiting for you to enter something. When you respond (and press ENTER), the computer puts what you typed in N\$. If you type

```
TONY (remember to press ENTER)
```

N\$ will contain the letters T, O, N, and Y. The \$ after N indicates that its value is text or typed characters (called a *string*) rather than a number. We refer to N\$ as a *string variable*.

An obvious question is, How does the user know what to enter? The question mark and cursor alone do not tell you whether the computer wants your name, your address, or any of a million other possibilities. To solve this problem, you should put a *prompt* in each INPUT. You must put the prompt in quotation marks immediately after the word INPUT and follow it with a colon.

For example, to ask for a name, we would have

```
100 INPUT "WHAT IS YOUR NAME?":N$
```

Type a colon here. Remember
? is on the front of the I
key. Press FCTN and I to
type it.

The computer prints

```
WHAT IS YOUR NAME?
```

as a prompt, followed by the cursor. Note that the screen stays green. Now you know how to respond. Let us see how this works in an actual program.

Program 4-1 Printing Your Name 20 Times

```
NEW
100 REM PROGRAM 4-1
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 FOR J=1 TO 20
150 PRINT N$
160 NEXT J
RUN
```

\$ is the shifted 4 key.

Do not put your name in the program. Wait until the computer asks you to enter it.

You will see the question

```
WHAT IS YOUR NAME?
```

followed by the cursor. Type your name and press ENTER. The screen now looks the same as after Program 2-3.

Note that you should always enter NEW before NUM. Otherwise, old lines will appear when the computer reaches their numbers in the sequence.

Type carefully. Remember to press SHIFT to enter :, -, and \$ but FCTN to enter " and ?. In particular, always think a second before typing \$. If you press FCTN and 4 instead of SHIFT and 4, you will clear the line you are typing, moving it up on the screen but not entering it into the computer's memory.

Program 4-2 Your Name Moving Right

```
NEW
100 REM PROGRAM 4-2
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 FOR C=1 TO 10
150 PRINT TAB(C);N$
160 NEXT C
RUN
```

Again you will see

```
WHAT IS YOUR NAME?
```

on the screen, followed by the cursor. You may want to put a space after the question mark to keep it from being crowded against the cursor. Type your name and press ENTER. The screen should look like this:

```

      TONY
    TONY
  TONY
TONY
  TONY
    TONY
      TONY
        TONY
          TONY

```

The nice feature of using INPUT is that we can type RUN again and enter a different name. The computer will then print that name without any change in the program. We can print any name just by typing it when

```
WHAT IS YOUR NAME?
```

appears on the screen. Run Program 4-2 and enter a friend's name when you see

```
WHAT IS YOUR NAME?
```

on the screen. You can see how handy INPUT is. Let's redo Program 2-9 and print a giant crossing pattern with two names.

Program 4-3 Input Names in a Crossing Pattern

```

NEW
100 REM PROGRAM 4-3
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 FOR C=1 TO 15
170 PRINT TAB(C);N$
180 PRINT TAB(16-C);NF$
190 NEXT C
RUN

```

The apostrophe is on the front of the O key. Press FCTN and O to type it.

We now have N\$ and NF\$. Be sure to press ENTER after each line.

The computer prints

```
WHAT IS YOUR NAME?
```

Type your name and press the ENTER key. The computer now prints

```
WHAT IS YOUR FRIEND'S NAME?
```

Type your friend's name and press ENTER. The computer does not care if you split the name between two lines. If, however, you want it all on one line, press the space bar until the cursor moves down a line. Leading spaces do not affect the string variable. You should get the same result as from Program 2-9. By using two INPUT statements, we can enter two different names and the computer will store them in N\$ and NF\$. We can then put N\$ and NF\$ wherever we want the names to appear. If we enter the names TONY and SALLY in response to Program 4-3, N\$ will be T, O, N, and Y, and NF\$ will be S, A, L, L, and Y.

N\$ → T O N Y
NF\$ → S A L L Y

To print two names at various places around the screen, we use a combination of PRINTs and TABs.

Program 4-4 Input Names Around the Screen

```
NEW
100 REM PROGRAM 4-4
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT
170 PRINT
180 PRINT TAB(13);N$
190 PRINT
200 PRINT TAB(5);NF$
210 PRINT
220 PRINT
230 PRINT
240 PRINT N$;NF$
RUN
```

Be sure to press ENTER after each line.

Place semicolons here.

If you type TONY and SALLY, your screen will look like this:

```

          TONY
    SALLY
TONYSALLY
```


We do not actually need a separate line for each PRINT. Furthermore, we can use a colon in a PRINT statement to skip down a line. So you could enter Program 4-4 as follows:

Program 4-5 Compacted Names All Over

```
NEW
100 REM PROGRAM 4-5
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT:;TAB(13);N$::TAB(5);NF$:::N$;NF$
RUN
```

Place semicolons here.

Place colons here.

The colon in a PRINT tells the computer to move to the next line. We can extend Program 4-4 to have the computer produce notes and greeting cards. For example, we could draw a heart and put a friend's name on it.

Program 4-6 Valentine's Day Card

```
NEW
100 REM PROGRAM 4-6
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 PRINT "  **  "
150 PRINT "* * * * *"
160 PRINT "* * * * *"
170 PRINT " *      * "
180 PRINT " *      * "
190 PRINT "  * *  "
200 PRINT "    *    "
210 PRINT "HAPPY VALENTINE'S"
220 PRINT "    DAY      "
230 PRINT N$
RUN
```

3 spaces

Be careful with lines that reach precisely to the right edge of the screen. Even though the cursor proceeds down to the left edge of the next empty line, you must still press ENTER to put the line in memory. Note that there is no prompt (>) on the empty line, so you cannot start typing the next statement.

To make a Christmas card, we could draw a tree with lights.

Program 4-7 Christmas Card

```
NEW
100 REM PROGRAM 4-7
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT "      "
170 PRINT "    * * "
180 PRINT "  *  O  *"
190 PRINT " *  O O  *"
200 PRINT "*  O O O  *"
210 PRINT "-----"
220 PRINT "  ! !  "
230 PRINT "  ---  "
240 PRINT "MERRY CHRISTMAS"
250 PRINT "AND HAPPY NEW YEAR"
260 PRINT TAB(5);NF$
270 PRINT TAB(5);"FROM"
280 PRINT TAB(5);N$
RUN
```

Diagram annotations for Program 4-7:

- An arrow points from the text "4 spaces" to the four spaces between the two asterisks on line 170.

Program 4-7 produces a strange effect. The tree seems to rise from the ground like a Halloween ghost or goblin. This happens because the computer prints at the bottom of the screen and moves the old lines up. Amusement parks create this effect by raising figures up through trap doors.

We can use the same approach to create other cards.

Program 4-8 New Year's Card

```
NEW
100 REM PROGRAM 4-8
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT "      /\      "
170 PRINT "    _/  \_      "
180 PRINT "  \  HAPPY  /      "
190 PRINT "   \  NEW  /      "
200 PRINT "    /  YEAR  \      "
210 PRINT "  /          \      "
220 PRINT "----\  /----"
230 PRINT "      \  /      "
240 PRINT " BEST WISHES"
250 PRINT "      TO      "
260 PRINT TAB(4);NF$
270 PRINT TAB(4);"FROM"
280 PRINT TAB(4);N$
RUN
```

Diagram annotations for Program 4-8:

- An arrow points from the text "4 spaces" to the four spaces between the backslash and forward slash on line 160.
- An arrow points from the text "\ is on the Z key. Press FCTN and Z to enter it." to the backslash character on line 170.
- An arrow points from the text "Underscore (_) is on the U key. Press FCTN and U to enter it." to the underscore character on line 170.

To make a birthday card, we could write a greeting or draw a cake with candles.

Program 4-9 Birthday Greeting

```
NEW
100 REM PROGRAM 4-9
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT "*****" ← 18 asterisks
170 PRINT "HAPPY BIRTHDAY TO"
180 PRINT NF$
190 PRINT "FROM YOUR FRIEND"
200 PRINT N$
210 PRINT "*****"
RUN
```

Since lines 160 and 210 are the same, you could shorten this program by defining BORDER\$ to be a string of 18 asterisks. We would then have

```
155 BORDER$="*****"
160 PRINT BORDER$
210 PRINT BORDER$
```

Program 4-10 Birthday Cake Card

```
NEW
100 REM PROGRAM 4-10
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT " /-----/" ← 3 spaces
170 PRINT " / 000000 /" ← 11 dashes
180 PRINT " / : : : : : /"
190 PRINT " !-----! "
200 PRINT " !HAPPY BIRTHDAY!"
210 PRINT " !-----!/" ← 12 underscores.
220 PRINT NF$ Press FCTN and U.
230 PRINT "FROM YOUR FRIEND,"
240 PRINT N$
RUN
```

For Halloween, we could draw a pumpkin and print both our name and a friend's name on the card. The pumpkin rising from the ground adds to the Halloween spirit.

Program 4-11 Halloween Card

```

NEW
100 REM PROGRAM 4-11
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 INPUT "WHAT IS YOUR FRIEND'S NAME?":NF$
150 CALL CLEAR
160 PRINT "DEAR ";NF$
170 PRINT
180 PRINT "(--!!--)"
190 PRINT "(  O O  )"
200 PRINT "(   X   )"
210 PRINT "(  =====  )"
220 PRINT "(-----)"
230 PRINT
240 PRINT "HAPPY HALLOWEEN"
250 PRINT "FROM YOUR FRIEND,"
260 PRINT N$
RUN

```

A Halloween card can also have a picture of a black cat, even though cats usually jump down from walls or fences rather than rise from the ground.

Program 4-12 Halloween Cat

```

NEW
100 REM PROGRAM 4-12
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 PRINT " ^---^ " ← ^ is shifted 6.
150 PRINT "/ * * \"
160 PRINT "\ ~~~ / " ← ~ is on the W key. Press FCTN and W
170 PRINT " ===== " to enter it.
180 PRINT "/ \ "
190 PRINT "\ / "
200 PRINT "/! ! !\" ← ! is on the A key. Press FCTN and A
210 PRINT "\! ! !/" to enter it.
220 PRINT " [~][~]" ← [ and ] are on the R and T keys,
230 PRINT respectively. Press FCTN and R to enter [ ;
240 PRINT " BOO " press FCTN and T to enter ].
250 PRINT "FROM YOUR FRIEND"
260 PRINT N$
RUN

```

For Mother's Day, we could create a screen with 100 I LOVE YOUs.

Program 4-13 Mother's Day Card

```
NEW
100 REM PROGRAM 4-13
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 PRINT "HAPPY MOTHER'S DAY, MOM"
150 FOR C=1 TO 100
160 PRINT "-I LOVE YOU-"; ← Be sure to put a semicolon here.
170 NEXT C
180 PRINT
190 PRINT "FROM YOUR DAUGHTER"
200 PRINT N$
RUN
```

As in Chapter 2, the screen eventually fills and all you see is a flickering bottom line. Note that the right side flickers more noticeably than the left side, since it is blank for a longer time.

For Easter, we could draw a giant cross.

Program 4-14 Easter Card

```
NEW
100 REM PROGRAM 4-14
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 PRINT "  ---  "
150 PRINT "   ||   "
160 PRINT "   ||   "
170 PRINT "----! !----"
180 PRINT "! PEACE !"
190 PRINT "----! !----"
200 PRINT "   ||   "
210 PRINT "   ||   "
220 PRINT "  ---  "
230 PRINT "HAPPY EASTER"
240 PRINT "    FROM    "
250 PRINT N$
RUN
```

And what would Easter be like without the Easter Bunny?

Program 4-15 Easter Bunny

```
NEW
100 REM PROGRAM 4-15
110 CALL CLEAR
120 INPUT "WHAT IS YOUR NAME?":N$
130 CALL CLEAR
140 PRINT "/-\ /-\\"
150 PRINT "|:|:|:"
160 PRINT "\ /-\ /\"
170 PRINT "/\ /-\ /\"
180 PRINT "|: * * |:"
190 PRINT "\ =~= /"
200 PRINT "/\ /\"
210 PRINT "|:|:|:"
220 PRINT "|:|:|:"
230 PRINT "|:|:|:"
240 PRINT "\ /\"
250 PRINT "(-) (-)"
260 PRINT
270 PRINT "HAPPY EASTER"
280 PRINT "FROM"
290 PRINT N$
RUN
```

Underscore (press FCTN and U)

~ is on the W key. Press FCTN and W to enter it.

Underscore (press FCTN and U)

The slowly rising Easter Bunny calls to mind an old amusement park where you can see (and hear) the mechanisms. Of course, the screen's changing from green to blue does not add to the realism either.

5

THE PERSONAL TOUCH

INPUT lets the computer converse with you almost as if it were a person. For example, it can ask for your name and weave it into a story.

Program 5-1 Short Story with Your Name

```
NEW
100 REM PROGRAM 5-1
110 CALL CLEAR
120 INPUT "ENTER YOUR NAME ":N$ ← Be sure to press SHIFT and 4 to
130 CALL CLEAR                                     type $, not FCTN and 4. FCTN
140 PRINT "SEE DICK RUN."                                     and 4 is CLEAR.
150 PRINT "SEE JANE RUN."
160 PRINT "SEE SPOT RUN."
170 PRINT "SEE ";N$;" RUN." ← Note the spaces after SEE and
180 PRINT "WATCH OUT, ";N$;"", "                                before RUN. What happens if you
190 PRINT "HERE COMES THE MAN"                                omit them?
200 PRINT "WHOSE WINDOW YOU BROKE."
210 PRINT "RUN FASTER, ";N$;"!"
RUN
```

! is the shifted 1 key.

The computer prints only 28 characters on a line. It then continues on the next line, starting with the 29th character. If you enter a long line, the computer will break it in an arbitrary place, making the output look strange. To keep this from happening, restrict all your lines to a maximum of 28 characters. Even then, the right-hand border will be uneven.

If you write your own story, be careful where you put the quotation marks. Look at lines 170, 180, and 210 as typical examples. All background words such as SEE, RUN, WATCH OUT, and RUN FASTER must have quotation marks around them. So also must punctuation marks such as commas, exclamation marks, and periods, as well as extra spaces that separate the name from the background words. The name N\$, however,

must not be inside quotation marks—this is difficult to see in lines with many quotation marks, so balance marks from left to right. There should always be an even number.

Here's another story about N\$ for you to try:

```
140 PRINT "LONG AGO, IN THE FAR AWAY"
150 PRINT "LAND OF OG, THERE LIVED THE"
160 PRINT "NOBLE ";N$;" ONE DAY,"
170 PRINT N$;" WENT OUT TO SLAY A"
180 PRINT "DRAGON. UNFORTUNATELY, THE"
190 PRINT "DRAGON SLEW ";N$;" INSTEAD."
200 PRINT "TOO BAD, ";N$;" BETTER"
210 PRINT "LUCK NEXT TIME!"
```

We can use as many INPUTs as we want to obtain information. The computer can then produce a letter or notice made up the way you want.

Program 5-2 Homework Assignment Notice

```
NEW
100 REM PROGRAM 5-2
110 CALL CLEAR
120 INPUT "ENTER YOUR NAME ";N$
130 INPUT "ENTER SUBJECT ";S$
140 INPUT "ENTER CHAPTER NUMBER ";C
150 CALL CLEAR
160 PRINT
170 PRINT "FROM: PROFESSOR ";N$
180 PRINT
190 PRINT "TO: ";S$;" CLASS"
200 PRINT
210 PRINT "WEEKLY ASSIGNMENT:"
220 PRINT "READ CHAPTER ";C
RUN
```

Try using colons to produce blank lines rather than PRINTs with nothing after them. This will shorten the program.

We can use INPUT for either numbers, such as C, or text, such as N\$ and S\$. The dollar sign at the end of a name indicates text. You may add blank lines, TABs, or borders to improve the notice's appearance. Watch the length of your lines. The notice will look fine if your name is SMITH, JOHNSON, or even PADEREWSKI and if your subject is ENGLISH, ELECTRONICS, or even AMERICAN HISTORY. You will run into trouble, however, if your name is IPPOLITOV-IVANOV or your subject is CHEMICAL ENGINEERING or ADVANCED BASKETWEAVING.

Program 5-3 Legal Notice

```
NEW
100 REM PROGRAM 5-3
110 CALL CLEAR
120 INPUT "ENTER PLAINTIFF'S NAME ";P$
130 INPUT "ENTER DEFENDANT'S NAME ";D$
140 INPUT "ENTER JUDGE'S NAME ";J$
150 CALL CLEAR
160 PRINT "OFFICIAL PUBLIC NOTICE"
170 PRINT ::"NEXT CASE IS:"
180 PRINT ::P$;" , PLAINTIFF"
190 PRINT ::"VERSES"
200 PRINT ::D$;" , DEFENDANT"
210 PRINT ::"BEFORE JUDGE ";J$
220 PRINT "*****"
RUN
```

A colon tells the computer to go to the start of the next line.

26 asterisks

We can call our strings anything we want. In practice, we generally use meaningful names such as P\$ for plaintiff, D\$ for defendant, and J\$ for judge. Of course, we can use INPUT to enter any text, not just people's names. In response to

```
INPUT "ENTER YOUR ADDRESS ":A$
```

we can enter any typed characters (letters, numbers, punctuation marks such as — or ., and so on). However, if the line is

```
INPUT "ENTER YOUR AGE ":A
```

we can only enter a number.

A\$ holds typed characters.

A holds only a number.

The TI-99/4A can serve as an inexpensive version of the elaborate message boards you often see in airports, stadiums, convention centers, and large hotels. The following program, for example, provides a news bulletin service.

Program 5-4 Extra! Extra! Read All About It!

```
NEW
100 REM PROGRAM 5-4
110 CALL CLEAR
120 INPUT "ENTER DATELINE ":D$
130 INPUT "ENTER LINE 1 ":L1$
140 INPUT "ENTER LINE 2 ":L2$
150 INPUT "ENTER LINE 3 ":L3$
160 INPUT "ENTER LINE 4 ":L4$
170 CALL CLEAR
180 PRINT TAB(2);"TI COMPUTER NEWS BULLETIN"
190 PRINT ::D$
200 PRINT ::L1$
210 PRINT ::L2$
220 PRINT ::L3$
230 PRINT ::L4$
RUN
```

Three colons in front results in three blank lines, since each one makes the computer go down a line.

You might try something like

DATELINE: WASHINGTON, D.C.
MARTIANS INVADE CITY!
TAKE OVER CONGRESS,
WHITE HOUSE, AND
SUPREME COURT!

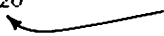
or

DATELINE: WASHINGTON, D.C.
 MARTIANS SURRENDER! MANY
 BORED TO DEATH IN CONGRESS!
 OTHERS SENTENCED TO SERVE
 ON PRESIDENTIAL COMMISSIONS!

There are a few things you must watch here (besides political sensitivities). In the first place, if your text includes commas or leading spaces that you want included, you will have to enclose them in quotation marks. For example, to enter WASHINGTON, D.C., you must type "WASHINGTON, D.C." If you forget the quotation marks, the computer will come back with

```
* WARNING
  INPUT ERROR IN 120
  TRY AGAIN:
```

Line number on INPUT



No damage is done, since you can now enter the line correctly. Be sure to enclose each text line in quotation marks to avoid having this happen.

Another problem is that the computer is a little slow moving down a line. In fact, it acts like a typewriter that must move its mechanical carriage. So if your entry extends beyond column 28 and the computer must continue it on the next line, you must wait for the cursor. If you don't, the computer will miss a character or two. Always watch the screen to be sure that the computer has accepted everything you typed.

Here, having the news bulletins move up the screen (and the screen turn from green to blue) adds to the tension and excitement. You could use this approach to create a computerized scoreboard for a Little League game or a computerized bulletin board for a store, classroom, or clubhouse.

We can also use INPUT to write a program that asks a series of questions. The computer can then serve as an interviewer for hospitals, clinics, government agencies, insurance offices, and other businesses or institutions that must collect vital statistics.

Program 5-5 Computer Interviewer

```
NEW
100 REM PROGRAM 5-5
110 CALL CLEAR
120 INPUT "ENTER YOUR FIRST NAME ":F$
130 INPUT "ENTER YOUR MIDDLE INITIAL ":M$
140 INPUT "ENTER YOUR LAST NAME ":L$
150 INPUT "ENTER YOUR STREET ADDRESS ":A$
160 INPUT "ENTER YOUR CITY ":C$
170 INPUT "ENTER YOUR STATE ":S$
180 INPUT "ENTER YOUR ZIP CODE ":Z$
190 CALL CLEAR
200 PRINT ::"CLIENT'S VITAL STATISTICS"
210 PRINT ::"NAME: ":L$;" ", "F$;" "M$;" ". "
220 PRINT "ADDRESS: ":A$;"C$;" ", "S$;" "Z$
RUN
```

Be sure to press SHIFT each time you want to type a colon. If you forget, you will get a semicolon instead. This will lead to an **INCORRECT STATEMENT** error in an INPUT.

Be careful to enter the colons and semi-colons in lines 210 and 220 correctly.




Be careful not to type too fast when entering your name or address, particularly when the computer is moving down a line on the screen. Also beware of addresses that contain a comma, such as 2150 MAIN ST., APT. 35. You must put quotation marks around them.

In practice, many people actually prefer a computer interviewer over a human interviewer, much as many people prefer to use an automatic teller machine even when a bank is open. One nice feature of the computer interviewer is its lack of human prejudices. It does not notice that a person has a strange appearance, is handicapped or disfigured, has language problems, or is slow to respond to questions.

Besides acting as interviewers, computers of the future will surely provide a wide range of expert assistance. While the following programs are humorous, they show the computer's inherent capabilities. Can you tell from the answers alone whether you are talking to a computer or a person?

Program 5-6 Doctor's Assistant

```
NEW
100 REM PROGRAM 5-6
110 CALL CLEAR
120 PRINT "DR. SMITH IS NOT IN. I AM"
130 PRINT "HIS COMPUTER ASSISTANT.":::
140 INPUT "TELL ME YOUR NAME ":N$
150 INPUT "TELL ME YOUR PROBLEM ":P$
160 CALL CLEAR
170 PRINT "DEAR ";N$;" : "
180 PRINT :::"I AM SORRY TO HEAR YOU HAVE"
190 PRINT P$;" . GO TO BED"
200 PRINT "EARLY, DRINK LIQUIDS, TAKE"
210 PRINT "TWO ASPIRIN, AND CALL DR."
220 PRINT "SMITH IN THE MORNING."
230 PRINT "I HOPE YOU FEEL BETTER SOON."
RUN
```

When you add colons at end of a line, you must add one more colon than the number of lines you want to skip.

The assistant does quite well if your problem is A COLD, A SORE THROAT, or A HEADACHE. It runs into grammatical and medical difficulties if your problem is A BROKEN ARM, A NAIL IN MY FOOT, or A COMPLAINT ABOUT MY BILL.

Program 5-7 Tax Collector's Assistant

```
NEW
100 REM PROGRAM 5-7
110 CALL CLEAR
120 PRINT "I AM YOUR FRIENDLY TAX"
130 PRINT "COLLECTOR'S ASSISTANT."
140 PRINT :::"WE ARE HERE TO HELP YOU."
150 INPUT "TELL ME YOUR NAME ":N$
160 PRINT "HOW MUCH DID YOU EARN"
170 INPUT "LAST YEAR? ":EARN
180 PRINT "HOW MUCH DID YOU REALLY EARN"
190 INPUT "LAST YEAR? ":EARN
200 CALL CLEAR
210 PRINT "YOU CANNOT FOOL ME, ";N$;" ! "
220 PRINT "I KNOW YOU MADE MORE THAN"
230 PRINT EARN;" LAST YEAR!"
240 PRINT "PAY YOUR TAX OR GO TO JAIL!"
250 PRINT "HAVE A NICE DAY, ";N$;" ."
RUN
```

Note that we had to split the questions HOW MUCH DID YOU EARN LAST YEAR? and HOW MUCH DID YOU REALLY EARN LAST YEAR? into two lines (a PRINT followed by an INPUT) because of their length.

As the programs become longer, you will surely find that you make more errors and that they are more difficult to correct. The following feature of the TI 99/4A can help you if you must correct several consecutive lines. Start in the usual way by pressing EDIT, followed by the lowest line number and ENTER. When you are finished with that line, however, you can enter it and move right on to the next higher-numbered line by pressing FCTN and X (the down arrow) simultaneously. Similarly, pressing FCTN and E (the up arrow) will display the next lower-numbered line. This is the main use of the up and down arrow keys; unfortunately, they will not move the cursor up or down a line when you are working on statements that occupy more than one line on the screen.

Programs 5-5 through 5-7 let the computer ask questions, read the answers, and repeat them like a parrot. However, it cannot determine if the answers are reasonable, correct, or formed properly. We need another BASIC statement that lets the computer compare things and decide what to do on the basis of the comparison. This statement is IF-THEN. We call the basis for the computer's decision a *condition*. The condition follows IF, and following THEN is the statement to do next if the condition is true. Typical examples of IF-THEN are

```
IF YEAR=1776 THEN 300
IF EXPLORER$="COLUMBUS" THEN 1200
IF N$=KING$ THEN 450
IF TEMP=FREEZING THEN 3000
```

A simple exercise shows how IF-THEN works.

Program 5-8 Simple Arithmetic Quiz

```
NEW
100 REM PROGRAM 5-8
110 CALL CLEAR
120 INPUT "WHAT IS 2+2? ":A
130 IF A=4 THEN 150
140 PRINT "SORRY, THE ANSWER IS 4"
150 END
RUN
```

Be sure to press SHIFT and = to type +; pressing FCTN and = (QUIT) will exit from BASIC and wipe out your program.

After you enter RUN, you will see the question

```
WHAT IS 2+2?
```

Answer by entering 4. The computer does not reach line 140 because A=4 (the condition) in line 130 is true, thus causing it to do line 150 next. We call this departure from the usual numerical sequence a *branch*. The END in line 150 simply concludes the program.

Now enter RUN again. This time enter 5 as your answer. You will see the message

SORRY, THE ANSWER IS 4

on the screen. Because $A=4$ is not true at line 130, the computer ignores THEN and proceeds normally to line 140.

A quiz with several questions requires a series of INPUT statements and IF-THENs to check the answers.

Program 5-9 Arithmetic Quiz with Two Questions

```
NEW
100 REM PROGRAM 5-9
110 CALL CLEAR
120 INPUT "WHAT IS 5+5? ":A
130 IF A=10 THEN 150
140 PRINT "WRONG, THE ANSWER IS 10"
150 INPUT "WHAT IS 3x4? ":B
160 IF B=12 THEN 180
170 PRINT "WRONG, THE ANSWER IS 12"
180 END
RUN
```

Be sure to press SHIFT to enter +, but FCTN to enter ?.

In line 120, the first INPUT, the computer calls the answer A. It then checks A against the correct answer. If A is not 10, the computer goes to the next line (line 140) and prints an error message. At line 150, the computer asks another question. Everything is the same, except that the answer is B instead of A. Now let's try questions where the answer is a name instead of a number.

Program 5-10 Geography Quiz

```
NEW
100 REM PROGRAM 5-10
110 CALL CLEAR
120 PRINT "WHAT IS THE CAPITAL"
130 INPUT " OF GEORGIA? ":C$
140 IF C$="ATLANTA" THEN 160
150 PRINT "WRONG, THE ANSWER IS ATLANTA"
160 END
RUN
```

We must split the question because of its length.

The answer is C\$ because it is text rather than a number.

We could even ask an easier question if the student misses the first one.

Program 5-11 Geography Quiz with an Easy Question

```
NEW
100 REM PROGRAM 5-11
110 CALL CLEAR
120 PRINT "WHAT IS THE CAPITAL"
130 INPUT " OF FRANCE? ":C$
140 IF C$="PARIS" THEN 220
150 PRINT "WRONG, THE ANSWER IS PARIS."
160 PRINT "LET'S TRY SOMETHING EASIER."
170 PRINT "IN WHAT CITY IS THE TOWER"
180 INPUT " OF LONDON? ":T$
190 IF T$="LONDON" THEN 220
200 PRINT "PLEASE TRY AGAIN."
210 GOTO 170
220 PRINT "THAT IS CORRECT."
RUN
```

In line 210, we introduced a minor BASIC statement called GOTO. This simply tells the computer to do the specified line number next. Thus, GOTO 170 means "do line 170 next." The computer then continues normally from line 170. If you think of the computer working its way down through a program like a person reading a newspaper or magazine article, then GOTO 170 is like the direction "continued on p. 170."

6

DRAWING STATIONARY OBJECTS

The computer lets us put a character anywhere on the screen with the single statement

`CALL HCHAR (ROW, COLUMN, CHARACTER)`

Row number (1 to 24) Column number (1 to 32) Character in the computer's internal code

Unfortunately, we must provide CHARACTER in the computer's own internal code. This code, called ASCII (pronounced *ass-kee*), is listed in Table 6-1; you may think of it as the computer equivalent of the dots and dashes in Morse code. For example, to put an asterisk in row 3 and column 16, we must first look up * in Table 6-1. Since its code is 42, the statement we need is

```
CALL HCHAR (3, 16, 42)
```

Be careful when using HCHAR. The row number must be between 1 and 24, and the column number must be between 1 and 32. Otherwise, you will get the error message

```
BAD VALUE IN
```

Note that the computer lets HCHAR put a character anywhere on a 32-column screen; it does not restrict you to the 28 columns it uses for text. However, the leftmost and rightmost columns (1 and 32) are so close to the edges of the screen that you will not be able to distinguish characters placed in them. So you should restrict HCHAR to columns 2 through 31.

TABLE 6-1 TI CHARACTER CODES

Symbol	Code	Symbol	Code	Symbol	Code
space	32	A	65	A	97
!	33	B	66	B	98
"	34	C	67	C	99
#	35	D	68	D	100
\$	36	E	69	E	101
%	37	F	70	F	102
&	38	G	71	G	103
'	39	H	72	H	104
(	40	I	73	I	105
)	41	J	74	J	106
*	42	K	75	K	107
+	43	L	76	L	108
,	44	M	77	M	109
-	45	N	78	N	110
"	46	O	79	O	111
/	47	P	80	P	112
0	48	Q	81	Q	113
1	49	R	82	R	114
2	50	S	83	S	115
3	51	T	84	T	116
4	52	U	85	U	117
5	53	V	86	V	118
6	54	W	87	W	119
7	55	X	88	X	120
8	56	Y	89	Y	121
9	57	Z	90	Z	122
:	58	[	91	{	123
;	59	\	92		124
<	60	]	93	}	125
=	61	^	94	~	126
>	62	_	95	DEL	127
?	63	.	96		
@	64				

The following simple program puts asterisks in five different positions around the screen.

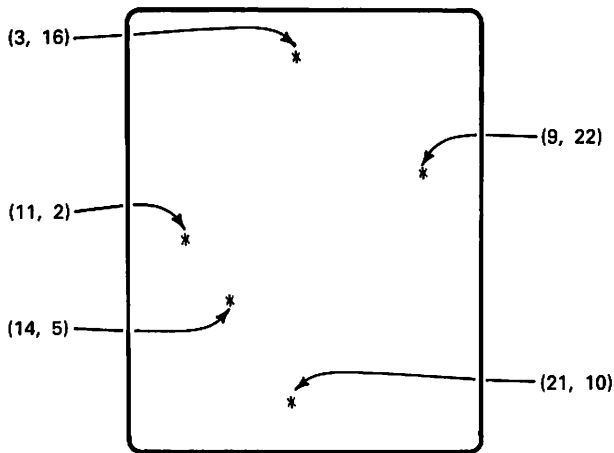
Program 6-1 Asterisks at Five Positions

```

NEW
100 REM PROGRAM 6-1
110 CALL CLEAR
120 CALL HCHAR(3,16,42)
130 CALL HCHAR(14,5,42)
140 CALL HCHAR(9,22,42)
150 CALL HCHAR(11,2,42)
160 CALL HCHAR(21,10,42)
RUN

```


The screen looks like this:



Experiment with the row numbers, column numbers, and character codes in Program 6-1. This will show you where the various positions are, as well as give you practice with the code table.

Looking up codes in Table 6-1 is a nuisance at best. We can let the computer do it for us by using the built-in ASC function. For example, we can set

```
AS=ASC ("*")
```

and then use AS instead of 42 in HCHAR statements. Note that the character must go inside both quotation marks and parentheses after ASC. The revised version of Program 6-1 is

```
110 CALL CLEAR
120 AS=ASC ("*")
130 CALL HCHAR (3, 16, AS)
140 CALL HCHAR (14, 5, AS)
150 CALL HCHAR (9, 22, AS)
160 CALL HCHAR (11, 2, AS)
170 CALL HCHAR (21, 10, AS)
```

In practice, naming characters and typing the ASC lines with their parentheses and quotation marks are just as much work as using Table 6-1.

Drawing a pair of eyes beginning in row 6 and column 10 requires several HCHARs. The eyes look like this:

```
(00)
  ↖ (Letter O, not zero)
```

The coded form from Table 6-1 is 40,79,79,41. The program to draw the eyes then becomes:

Program 6-2 Stationary Eyes

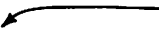
```
NEW
100 REM PROGRAM 6-2
110 CALL CLEAR
115 REM DRAW EYES IN A ROW (00)
120 CALL HCHAR(6,10,40)
130 CALL HCHAR(6,11,79)
140 CALL HCHAR(6,12,79)
150 CALL HCHAR(6,13,41)
RUN
```

Obviously, you must draw the picture on paper first and describe it in REM lines, since the HCHARs are almost impossible to visualize.

Program 6-2 uses four HCHARs to put four characters on the screen. We can shorten it a little by using HCHAR's ability to repeat a character horizontally. That is, we can combine lines 130 and 140 into the single statement

```
130 CALL HCHAR(6,11,79,2)
```

How many to print in a row



Program 6-3 Stationary Eyes, Alternate Form

```
NEW
100 REM PROGRAM 6-3
110 CALL CLEAR
115 REM DRAW EYES IN A ROW (00)
120 CALL HCHAR(6,10,40)
130 CALL HCHAR(6,11,79,2)
140 CALL HCHAR(6,13,41)
RUN
```

Here we used HCHAR's general form,

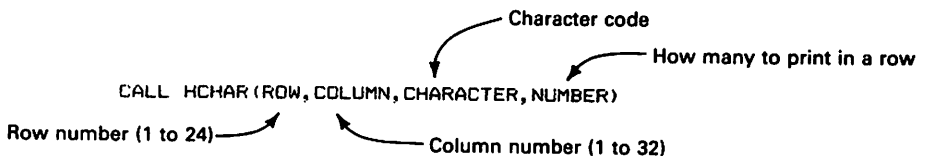
CALL HCHAR(ROW,COLUMN,CHARACTER,NUMBER)

Character code

How many to print in a row

Row number (1 to 24)

Column number (1 to 32)



For example, to print four asterisks in a row, we can use either four separate

```
CALL HCHAR(R,C,42)
```

statements or a single

```
CALL HCHAR(R,C,42,4).
```

Drawing larger objects involving many different characters requires many CALL HCHARs. To prevent mistakes:

1. Always draw the object on paper first.
2. Look up the code for each character in Table 6-1.
3. Write the HCHAR statements.

Let us now draw an alien creature that looks like this:

```
\-/  
(O)  
/-\
```

We start the picture in row 6 and column 13. The codes are

ALIEN PICTURE			CODES FOR PICTURE			
Row	#6	→ \ - /	92	45	47	← #6
	#7	→ (O)	40	79	41	← #7
	#8	→ / - \	47	45	92	← #8
		↑ ↑ ↓	↑ ↑ ↓			
		13 14 15	13 14 15			
		Column	Column			

Program 6-4 Stationary Alien

```
NEW  
100 REM PROGRAM 6-4  
110 CALL CLEAR  
115 REM DRAW THE FIRST ROW     \-/  
120 CALL HCHAR(6,13,92)  
130 CALL HCHAR(6,14,45)  
140 CALL HCHAR(6,15,47)  
145 REM DRAW THE SECOND ROW     (O)  
150 CALL HCHAR(7,13,40)  
160 CALL HCHAR(7,14,79)  
170 CALL HCHAR(7,15,41)  
175 REM DRAW THE THIRD ROW     /-\  
180 CALL HCHAR(8,13,47)  
190 CALL HCHAR(8,14,45)  
200 CALL HCHAR(8,15,92)  
RUN
```

The computer places the characters directly at the positions specified in the HCHARs; it does not print at the bottom of the screen as it does for a PRINT statement. Thus it draws the alien from left to right and top to bottom. Of course, you can change the direction by reordering lines 120 through 200.

One problem is that the computer distorts the picture, moving it up the screen, when it prints DONE and restores the cursor. To keep this from happening, add the line

```
210 GOTO 210
```

The computer will stay on this line forever, since it branches to itself. You must press FCTN and 4 (CLEAR) to regain control. Don't just touch these keys; hold them down until the program stops running.

To draw a bug, we use the picture

```

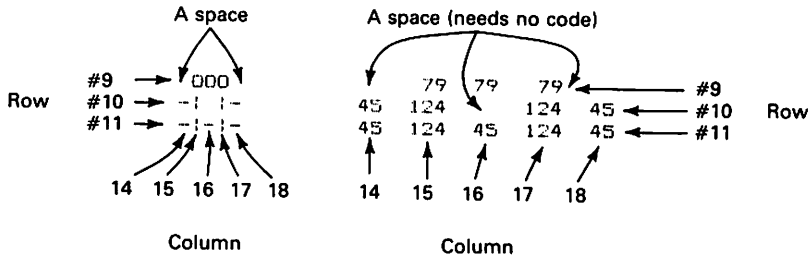
 000
-! !-
-! !-

```

The drawing contains the following distinct characters:

CHARACTER	CODE
0	79
!	124
-	45

Let us start the bug in row 9 and column 14. To convert it into HCHARs, we must put the codes in their proper screen positions:



Program 6-5 Stationary Bug

```

NEW
100 REM PROGRAM 6-5
110 CALL CLEAR
115 REM DRAW FIRST ROW      000
120 CALL HCHAR(9,15,79,3)
125 REM DRAW SECOND ROW    -! !-
130 CALL HCHAR(10,14,45)
140 CALL HCHAR(10,15,124)
150 CALL HCHAR(10,17,124)
160 CALL HCHAR(10,18,45)
165 REM DRAW THIRD ROW    -! !-
170 CALL HCHAR(11,14,45)
180 CALL HCHAR(11,15,124)
190 CALL HCHAR(11,16,45)
200 CALL HCHAR(11,17,124)
210 CALL HCHAR(11,18,45)
RUN

```

CALL HCHAR lets us put the same character in consecutive horizontal (row) positions. Unfortunately, the bug in Program 6-5 has duplicate characters vertically rather than horizontally. These pairs are

```

- , | , | , and - .
- | | |

```

To print these, we use VCHAR instead of HCHAR. VCHAR is just like HCHAR except that it prints vertically instead of horizontally. VCHAR's general form is

CALL VCHAR(ROW, COLUMN, CHARACTER, NUMBER)

When NUMBER = 1, VCHAR and HCHAR give the same result.

We can now rewrite Program 6-5 using a combination of HCHARs and VCHARs. We use VCHAR to draw the left legs, the left trunk, the right trunk, and the right legs. We use HCHAR to draw the eyes and the bottom trunk (a single dash, -). The picture thus reduces to the following statements:

```

CALL HCHAR(9, 15, 79, 3) → 000
CALL VCHAR(10, 14, 45, 2) → -| | - ← CALL VCHAR(10, 18, 45, 2)
                             -| | -
CALL VCHAR(10, 15, 124, 2) → ↑ ← CALL VCHAR(10, 17, 124, 2)
                             |
CALL HCHAR(11, 16, 45)      → -

```

Program 6-6 Stationary Bug, Short Form

```

NEW
100 REM PROGRAM 6-6
110 CALL CLEAR
115 REM DRAW EYES      000
120 CALL HCHAR(9, 15, 79, 3)
125 REM DRAW LEFT LEGS -
130 CALL VCHAR(10, 14, 45, 2)
135 REM DRAW LEFT SIDE OF BODY  |
140 CALL VCHAR(10, 15, 124, 2)
145 REM DRAW RIGHT SIDE OF BODY |
150 CALL VCHAR(10, 17, 124, 2)
155 REM DRAW RIGHT LEGS  -
160 CALL VCHAR(10, 18, 45, 2)
165 REM DRAW BOTTOM TRUNK -
170 CALL HCHAR(11, 16, 45)
RUN

```

Program 6-6 involves only six CALLs, instead of the ten in Program 6-5. As with HCHAR, you must be sure that VCHAR's row number is between 1 and 24 and its column number is between 1 and 32. Otherwise, you will get an error message and the program will not work.

7

MAKING PICTURES MOVE

We will now make our pictures move. To move something across the screen, we use a FOR-NEXT loop with HCHARs and VCHARs. Program 7-1 makes an asterisk move right.

Program 7-1 Asterisk Moving Right, without Erasure

```
NEW
100 REM PROGRAM 7-1
110 CALL CLEAR
120 FOR C=1 TO 20
130 CALL HCHAR(5,C,42)
140 NEXT C
150 RUN
```

Moves asterisk from column 1 to column 20

Asterisk moves in row 5.

An entire line of asterisks appears in a flash, extending from left to right. To produce a single moving object instead, we can use CALL CLEAR to erase the last asterisk before printing a new one.

Program 7-2 Asterisk Moving Right, with Erasure

```
NEW
100 REM PROGRAM 7-2
110 CALL CLEAR
120 FOR C=1 TO 20
130 CALL CLEAR
140 CALL HCHAR(5,C,42)
150 FOR K=1 TO 100
160 NEXT K
170 NEXT C
RUN
```

Slows the computer down

Now we see only a single asterisk floating across the screen like a balloon. To speed it up, change the upper limit in the tight loop (line 150) to 25 or 50.

The problem with using CALL CLEAR to erase old objects is that it erases the entire screen. It would therefore erase background scenery (such as cannons, tanks, targets, or planets) as well as the moving object. To erase just the moving object, we add another HCHAR that prints a space (code 32, from Table 6-1) over it.

Program 7-3 Asterisk Moving Right, Alternate Method

```

NEW
100 REM PROGRAM 7-3
110 CALL CLEAR
120 FOR C=1 TO 20
130 CALL HCHAR(5,C,42) ← Asterisk moves in row 5.
140 FOR K=1 TO 50 ← Slows the computer down
150 NEXT K
160 CALL HCHAR(5,C,32) ← Prints a space (code 32) over the
170 NEXT C               asterisk, erasing it.
RUN

```

The motion is somewhat jerky, since the columns are rather far apart. To move the asterisk left, we simply change the FOR loop to

```
120 FOR C=20 TO 1 STEP -1
```

Program 7-4 Asterisk Moving Left, without Shadow

Do not type NEW, so that Program 7-3 stays in memory.

```

100 REM PROGRAM 7-4
120 FOR C=20 TO 1 STEP -1
RUN

```

To move the eyes from Program 6-3, we use the same technique with a larger picture. However, to erase the eyes, we need CALL HCHAR(5,C,32,4) because the eyes are four columns wide. This HCHAR puts four spaces over the picture, erasing all of it.

Program 7-5 Eyes Moving Right

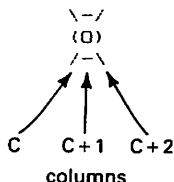
```

NEW
100 REM PROGRAM 7-5
110 CALL CLEAR
120 FOR C=1 TO 20
130 CALL HCHAR(5,C,40)
140 CALL HCHAR(5,C+1,79,2) ← Be sure to press SHIFT, not FCTN, to
150 CALL HCHAR(5,C+3,41)    type +. Remember, FCTN and + is QUIT.
160 CALL HCHAR(5,C,32,4) ← Prints spaces over the 4-column wide
170 NEXT C                  eyes
RUN

```

The eyes flicker noticeably as they move, since we are erasing the entire picture each time.

To move a picture that is more than one line high, we must remember to erase it completely. The alien from Program 6-4 begins in column C and looks like this:



It occupies 3 columns in rows 6, 7, and 8. To erase it, we need

```

CALL HCHAR(6,C,32,3)
CALL HCHAR(7,C,32,3)
CALL HCHAR(8,C,32,3)

```

or

```

FOR R=6 TO 8
CALL HCHAR(R,C,32,3)
NEXT R

```

in which C is the loop variable that moves the alien.

Program 7-6 Alien Moving Right

```

NEW
100 REM PROGRAM 7-6
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW ALIEN IN COLUMN C
126 REM DRAW TOP LINE          \-/\
130 CALL HCHAR(6,C,92)
140 CALL HCHAR(6,C+1,45)
150 CALL HCHAR(6,C+2,47)
155 REM DRAW MIDDLE LINE      (O)
160 CALL HCHAR(7,C,40)
170 CALL HCHAR(7,C+1,79)
180 CALL HCHAR(7,C+2,41)
185 REM DRAW BOTTOM LINE      /-\/
190 CALL HCHAR(8,C,47)
200 CALL HCHAR(8,C+1,45)
210 CALL HCHAR(8,C+2,92)
215 REM ERASE ALIEN
220 CALL HCHAR(6,C,32,3)
230 CALL HCHAR(7,C,32,3)
240 CALL HCHAR(8,C,32,3)
250 NEXT C
RUN

```

To avoid tempting fate by typing + repeatedly, you may want to insert

```

127 C1=C+1
128 C2=C+2

```

You can then use C1 instead of C+1 on lines 140, 170, and 200 and C2 instead of C+2 on lines 150, 180, and 210.

The motion looks strange, to say the least. Rather than moving continuously, the alien keeps appearing and disappearing. The top line appears first, then the middle line,

then the bottom line; then the entire picture disappears and starts appearing again to the right. It looks as though the alien were drifting along while someone pulled a window shade up and down rapidly in front of it.

The problem is the complete erasure and redrawing. In fact, a moving object does not vanish and then reappear in its new position. Instead, it moves continuously; the next time we see it, each part of it has moved a short distance.

We can produce more natural motion by making a column of spaces follow the picture. When the picture moves right one column, the column of spaces overprints its leftmost column. The result is that the picture moves right without ever disappearing. The easiest way to revise Program 7-6 is by putting the spaces in column C-1 and adjusting the FOR loop as follows:

Program 7-7 Alien Moving Right, with Spaces

Do not type NEW, so that Program 7-6 remains in memory.

```
100 REM PROGRAM 7-7
120 FOR C=2 TO 21
127 CALL VCHAR(5,C-1,32,30)

220 ←
230 ←
240 ←
RUN
```

These lines printed spaces over the alien.

The alien no longer appears and disappears. Instead, it flickers across the screen in sections. You might think it had a muscular or nervous disorder.

To draw a car, we use the picture

```
00
--
< **
--
00
```

The following HCHAR statements produce the drawing:

```
CALL HCHAR(10,C,79,2)    00
CALL HCHAR(11,C,45,2)    --
CALL HCHAR(12,C,42,2)    **
CALL HCHAR(13,C,45,2)    --
CALL HCHAR(14,C,79,2)    00
CALL HCHAR(12,C+2,62)    >
```

Note that the back of the car begins in column C (so it can be moved) and the nose of the car (>) is two columns farther right (C+2). Since the car is 5 rows tall, the statement to put a column of spaces behind it is

```
CALL VCHAR(10,C-1,32,5)
```

Program 7-8 Car Moving Right

```
100 REM PROGRAM 7-8
110 CALL CLEAR
115 REM MOVE CAR
120 FOR C=2 TO 21
125 REM DRAW CAR
130 CALL VCHAR(10,C-1,32,5) ← Erases shadow
140 CALL HCHAR(10,C,79,2)
150 CALL HCHAR(11,C,45,2)
160 CALL HCHAR(12,C,42,2)
170 CALL HCHAR(13,C,45,2)
180 CALL HCHAR(14,C,79,2)
190 CALL HCHAR(12,C+2,62)
200 NEXT C
RUN
```

The car appears to contract and then expand like an accordion. See how it looks if you overwrite it with spaces each time rather than carrying the spaces along with the drawing. To erase the entire car, use

```
FOR R=10 TO 14
CALL HCHAR(R,C,32,3)
NEXT R
```

If you make many changes in a program or add more lines (such as extra tight loops to slow down the computer), you will end up with highly irregular line numbers. In fact, in some sections you may find yourself with no spare line numbers for additions. You could, of course, renumber some lines yourself, but this can cause problems if the program contains GOTOs. Luckily, the TI-99/4A will renumber the entire program for you, taking care of all GOTOs automatically. All you must do is enter

RES

(*resequence*). Like NUM, you can follow RES with a starting line number, a comma, and a line spacing. Also as with NUM, the default values are 100 and 10. That is, typing RES alone will make the computer renumber your entire program, starting with line 100 and continuing with a spacing of 10. Note that RES rennumbers the entire program; you cannot renumber just part of it.

8

SOME THINGS MOVE WHILE OTHERS REMAIN STATIONARY

In Chapter 7, we made entire pictures move across the screen. To make parts of a picture move while other parts remain stationary, we must draw the stationary parts first using HCHAR or VCHAR and then draw and erase the moving parts in loops. We can then make bullets, rockets, cars, or bugs move against a stationary background. For example, let's first move a rocket right. The rocket looks like this:

```
|=>
```

It consists of character codes 32 (a space), 124, 61, and 62.

Program 8-1 Small Rocket Moving Right

```
NEW
100 REM PROGRAM 8-1
110 CALL CLEAR
120 FOR C=1 TO 20
130 CALL HCHAR(4,C,124)
140 CALL HCHAR(4,C+1,61)
150 CALL HCHAR(4,C+2,62)
160 NEXT HCHAR(4,C,32,3)
170 NEXT C
RUN
```

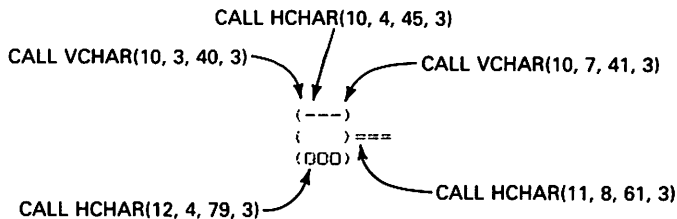
Now let's make a stationary tank fire the rocket. The tank looks like this:

```
(---)
(   )===
(000)
```

We make it start in the third column of the 10th row. The codes we need are

Character	Code
(	40
-	45
)	41
=	61
0	79

So the drawing consists of the following HCHAR and VCHAR statements:



Note that the tank's turret starts in the 8th column of the 11th row. The end of the barrel, position 11,11, is the rocket's starting position.

Program 8-2 Tank Firing a Rocket to the Right

```

NEW
100 REM PROGRAM 8-2
110 CALL CLEAR
115 REM DRAW TANK
120 CALL VCHAR(10,3,40,3)
130 CALL HCHAR(10,4,45,3)
140 CALL VCHAR(10,7,41,3)
150 CALL HCHAR(12,4,79,3)
160 CALL HCHAR(11,8,61,3)
165 REM FIRE ROCKET STARTING AT POSITION 11,9
170 FOR C=11 TO 29
180 CALL HCHAR(11,C,124)
190 CALL HCHAR(11,C+1,61)
200 CALL HCHAR(11,C+2,62)
205 REM ERASE ROCKET
210 CALL HCHAR(11,C,32,3)
220 NEXT C
RUN
  
```

Be careful to distinguish between VCHAR (lines 120 and 140) and HCHAR (lines 130, 150, and 160).

← Rocket is in the 11th row.

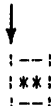
The tank stands still while the rocket moves right. The key here is that the rocket statements contain C, while the tank statements do not. The C loop starts at 11 and goes to 29. This makes the computer draw the rocket first in column 11 (the end of the barrel), then in 12, and so on, until it reaches 29. We can determine the initial C value either by counting characters or by experimenting.

A different scene is an arrow shot at a target, since here the moving object travels toward the stationary object rather than away from it.

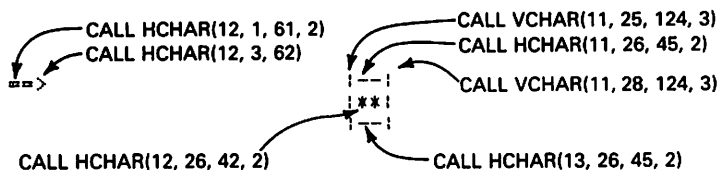
Place arrow here and move it.



Place target here and leave it.



With the top of the target in position 11,25, the HCHAR and VCHAR instructions become



Program 8-3 Arrow Shot at a Target

```
NEW
100 REM PROGRAM 8-3
110 CALL CLEAR
115 REM DRAW TARGET
120 CALL VCHAR(11,25,124,3)
130 CALL HCHAR(11,26,45,2)
140 CALL HCHAR(12,26,42,2)
150 CALL HCHAR(13,26,45,2)
160 CALL VCHAR(11,28,124,3)
165 REM MOVE ARROW LEFT TO RIGHT
170 FOR C=10 TO 21
180 CALL HCHAR(12,C,61,2)
190 CALL HCHAR(12,C+2,62)
195 REM ERASE ARROW
200 CALL HCHAR(12,C,32,3)
210 NEXT C
220 CALL CLEAR
RUN
```

We want to move the arrow from column 10 to column 21. We cannot go beyond 21 since the arrow occupies 3 columns.

We put an extra CALL CLEAR at the end to erase the final picture.

Let's draw a simple ball consisting of a single 0 (zero). To make it move, the computer must print it, overprint it with a space, and then print it in a different position. Only one ball should ever be on the screen at a time.

Program 8-4 Moving Ball

```
NEW
100 REM PROGRAM 8-4
110 CALL CLEAR
120 FOR C=1 TO 30
125 REM DRAW BALL
130 CALL HCHAR(7,C,48)
135 REM OVERPRINT BALL WITH SPACE
140 CALL HCHAR(7,C,32)
150 NEXT C
160 CALL CLEAR
RUN
```

Now let us add a stationary stick figure who could be throwing the ball. We can position the figure with

```
CALL HCHAR(5, 18, 46) → O
CALL VCHAR(6, 18, 124, 3) → |
CALL HCHAR(7, 17, 47) → /:\ ← CALL HCHAR(7, 19, 92)
                           |
CALL HCHAR(9, 17, 41) → ) ) ← CALL HCHAR(9, 19, 41)
```

Program 8-5 Stick Figure Throwing Ball

```
NEW
100 REM PROGRAM 8-5
110 CALL CLEAR
115 REM DRAW STICK FIGURE
120 CALL HCHAR(5,18,46)
130 CALL VCHAR(6,18,124,3)
140 CALL HCHAR(7,17,47)
150 CALL HCHAR(7,19,92)
160 CALL HCHAR(9,17,41)
170 CALL HCHAR(9,19,41)
175 REM MOVE BALL LEFT TO RIGHT
180 FOR C=20 TO 28
190 CALL HCHAR(7,C,48)
195 REM ERASE BALL
200 CALL HCHAR(7,C,32)
210 NEXT C
220 CALL CLEAR
RUN
```

The stick figure stands still while the ball moves. We can make the ball move toward the figure by changing line 180 to

```
180 FOR C=28 TO 20 STEP -1
```

You may want to change the figure to improve its form and give it more personality.

We can easily add some ground with the underscore symbol (on the front of the U key; press FCTN and U to type it) and a wall. Now the stick figure can be practicing handball.

Program 8-6 Stick Figure Playing Handball

```
NEW
100 REM PROGRAM 8-6
110 CALL CLEAR
115 REM DRAW STICK FIGURE
120 CALL HCHAR(5,18,46)
130 CALL VCHAR(6,18,124,3)
140 CALL HCHAR(7,17,47)
150 CALL HCHAR(7,19,92)
160 CALL HCHAR(9,17,41)
170 CALL HCHAR(9,19,41)
175 REM DRAW GROUND
180 CALL HCHAR(10,1,95,28)
185 REM DRAW WALL
190 CALL VCHAR(11,29,124,10)
```

```

195 REM MOVE BALL RIGHT
200 FOR C=20 TO 28
210 CALL HCHAR(7,C,48)
215 REM ERASE BALL
220 CALL HCHAR(7,C,32)
230 NEXT C
235 REM MOVE BALL LEFT
240 FOR C=28 TO 20 STEP -1
245 REM DRAW BALL
250 CALL HCHAR(7,C,48)
255 REM ERASE BALL
260 CALL HCHAR(7,C,32)
270 NEXT C
280 CALL CLEAR
RUN

```

This scene should remind you of the older arcade games. We must make the moving object rather small for its motion to look believable.

A somewhat different scene is a space station at the bottom of the screen firing a small missile upward. The station's HCHAR and VCHAR commands are

```

CALL HCHAR(22, 14, 45, 5)
CALL VCHAR(22, 13, 124, 2)
CALL HCHAR(23, 14, 45, 5)
CALL HCHAR(22, 16, 94)
CALL VCHAR(22, 19, 124, 2)

```

Note that we first draw the sides, top, and bottom and then put the station's launch silo (^) in the middle. To move the missile up the screen, we use

```

FOR R=21 TO 1 STEP -1
CALL HCHAR(R, 16, 42)
CALL HCHAR(R, 16, 32)
NEXT R

```

Program 8-7 Space Station Firing a Small Missile

```

NEW
100 REM PROGRAM 8-7
110 CALL CLEAR
115 REM DRAW SPACE STATION
120 CALL VCHAR(22, 13, 124, 2)
130 CALL VCHAR(22, 19, 124, 2)
140 CALL HCHAR(22, 14, 45, 5)
150 CALL HCHAR(23, 14, 45, 5)
160 CALL HCHAR(22, 16, 94)
165 REM FIRE MISSILE
170 FOR R=21 TO 1 STEP -1
180 CALL HCHAR(R, 16, 42)
190 CALL HCHAR(R, 16, 32)
200 NEXT R
210 CALL CLEAR
RUN

```

To have the space station fire missiles continuously, add

```
210 GOTO 170
```

We could draw a more complex missile like this:

```
Line 1      ^  
Line 2      :
```

Program 8-8 Space Station Firing a Large Missile

```
NEW  
100 REM PROGRAM 8-8  
110 CALL CLEAR  
115 REM DRAW SPACE STATION  
120 CALL VCHAR(22,13,124,2)  
130 CALL VCHAR(22,19,124,2)  
140 CALL HCHAR(22,14,45,5)  
150 CALL HCHAR(23,14,45,5)  
160 CALL HCHAR(22,16,94)  
165 REM FIRE MISSILE  
170 FOR R=19 TO 1 STEP -1  
175 REM DRAW MISSILE  
180 CALL HCHAR(R,16,94) ← Draw top of missile ^ .  
190 CALL HCHAR(R+1,16,124) ← Draw bottom of missile : .  
195 REM ERASE MISSILE  
200 CALL VCHAR(R,16,32,2) ← Overprint missile with 2 spaces.  
210 NEXT R  
220 CALL CLEAR  
RUN
```


9

MAKING CREATURES WALK

To make a figure walk, we must draw several pictures of it, each in a different stage of walking. This is, after all, how motion pictures work. A movie is really just a series of still pictures separated in time. Let's look at three pictures of a bug.

Picture #1

```
  |---\
  |  O |
  |---<
  |   /
  |   /
  |   /
```

Picture #2

```
  |---\
  |  O |
  |---<
  |   /
  |   /
  |   /
```

Picture #3

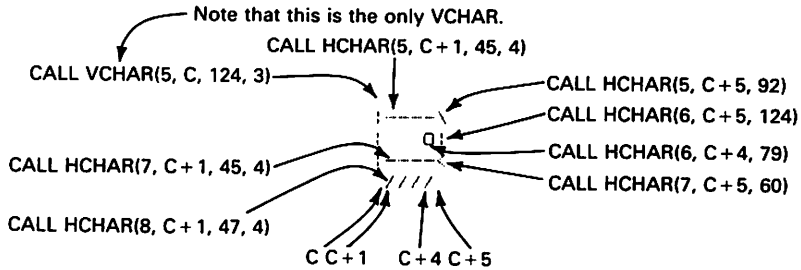
```
  |---\
  |  O |
  |---<
  |   /
  |   /
  |   /
```

If we draw the three pictures in the same place, the bug's legs appear to move. We can then move the pictures to the right, and the bug appears to walk. To make picture drawing easier, we introduce two new BASIC statements.

GOSUB AND RETURN STATEMENTS

GOSUB sends the computer to another part of the program. RETURN tells it to go back to where it came from. Every GOSUB needs a RETURN and every RETURN a GOSUB. Think of the computer as a person reading a book. At a point on the page, the book says, "Look at the map on page 247." The reader marks the current page, turns to page 247, examines the map, and returns to where he or she left off. The reader then continues as if no interruption had occurred. That is how GOSUB and RETURN work. Let us now see how we use them in programs. We generally refer to the secondary part of the program (reached via a GOSUB) as a *subroutine*.

The HCHARs and VCHARs needed to draw the bug are



The bug's body is the same in all three pictures; only the legs change. We draw the legs three different ways in the same place to make them appear to move.

Program 9-1 Stationary Bug with Moving Legs

```

NEW
100 REM PROGRAM 9-1
110 CALL CLEAR
115 REM DRAW BUG'S BODY
120 GOSUB 1000
130 FOR J=1 TO 20
135 REM DRAW LEGS
140 CALL HCHAR(8,4,47,4) ← Legs bent straight back /////.
150 CALL HCHAR(8,4,124,4) ← Straight legs ! ! ! !
160 CALL HCHAR(8,4,40,4) ← Legs bent part way forward ( ( ( ( .
170 NEXT J
180 GOTO 9000
995 REM DRAW PICTURE OF BUG'S BODY
1000 CALL VCHAR(5,3,124,3) ← Left side, the only VCHAR
1010 CALL HCHAR(5,4,45,4) ← Top
1020 CALL HCHAR(5,8,92) ← Forehead
1030 CALL HCHAR(6,8,124) ← Nose
1040 CALL HCHAR(7,8,60) ← Mouth
1050 CALL HCHAR(6,7,79) ← Eye
1060 CALL HCHAR(7,4,45,4) ← Bottom
1070 RETURN
9000 CALL CLEAR
RUN

```

The HCHARs on lines 140, 150, and 160 all draw the bug's legs in the same place. Since the body is on lines 5 through 7, the legs are always on line 8.

When you run Program 9-1, the legs move too rapidly; the effect is like a cartoon that has been speeded up too much. To slow the motion, insert

```

145 FOR K=1 TO 50
146 NEXT K

155 FOR K=1 TO 50
156 NEXT K

165 FOR K= 1 TO 50
166 NEXT K

```

You can use RES to renumber the lines, but note that it will put the subroutine and line 9000 right after the main program.

At line 120, GOSUB 1000 sends the computer to line 1000. It proceeds sequentially until it reaches the RETURN in 1070. The computer then takes up where it left off—that is, right after GOSUB 1000 (line 130). It then proceeds sequentially through the rest of the program.

To make the bug walk, we simply move its body right, drawing three pictures of its legs in each place. We must use a second subroutine to erase each picture and prevent a shadow.

Program 9-2 Bug Walking Right

```

NEW
100 REM PROGRAM 9-2
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW BUG'S BODY
130 GOSUB 1000
135 REM DRAW LEGS
140 CALL HCHAR(B,C+1,47,4)
150 FOR K=1 TO 20
160 NEXT K
170 CALL HCHAR(B,C+1,124,4)
180 FOR K=1 TO 20
190 NEXT K
200 CALL HCHAR(B,C+1,40,4)
210 FOR K=1 TO 20
220 NEXT K
225 REM ERASE OLD BUG
230 GOSUB 2000
240 NEXT C
250 GOTO 9000
995 REM PICTURE OF BUG
1000 CALL VCHAR(5,C,124,3)
1010 CALL HCHAR(5,C+1,45,4)
1020 CALL HCHAR(5,C+5,92)
1030 CALL HCHAR(6,C+5,124)
1040 CALL HCHAR(7,C+5,60)
1050 CALL HCHAR(6,C+4,79)
1060 CALL HCHAR(7,C+1,45,4)
1070 RETURN
1995 REM ERASE PICTURE OF BUG
2000 FOR R=5 TO 8
2010 CALL HCHAR(R,C,32,6)
2020 NEXT R
2030 RETURN
9000 CALL CLEAR
RUN

```

Enter NUM 1000 before starting the subroutine.

Left side, the only VCHAR

Enter NUM 2000 before starting the erasure subroutine.

The computer draws the bug's body and three pictures of its legs, one after another, at the same place; then it erases the entire bug, advances to the next position, and starts over. The subroutine beginning at line 2000 prints four rows of six spaces on top of the bug, erasing it completely.

We can also make the bug move left. All we must do is change the FOR statement (line 120) and put the face on the left side.

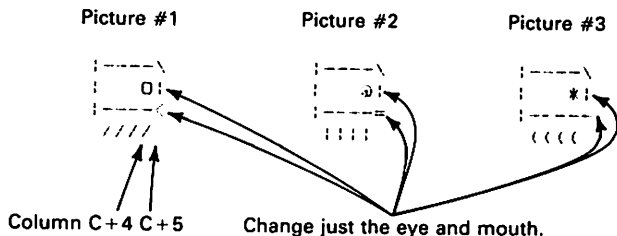
Program 9-3 Bug Walking Left

Do not type NEW. Leave Program 9-2 in memory.

```
100 REM PROGRAM 9-3
120 FOR C=20 TO 1 STEP -1
1000 CALL VCHAR(5,C+5,124,3) ← Right side
1020 CALL HCHAR(5,C,92) ← Forehead
1030 CALL HCHAR(6,C,124) ← Nose
1040 CALL HCHAR(7,C,60) ← Mouth
1050 CALL HCHAR(6,C+1,79) ← Eye
RUN
```

To change the subroutine lines, enter NUM 1000 and then edit each line as it appears. To leave line 1010 unchanged, simply press ENTER when you reach it. When line 1060 appears, press FCTN and 4 (CLEAR) to exit from the number mode. Be very careful here when you type +. If you are using FCTN to move the cursor, you could easily press FCTN and = instead of SHIFT and = and lose your entire program.

To make the bug's eyes blink and its mouth open and close, all we must change is the drawing of the eye and mouth. The three pictures become



Program 9-4 contains the revisions. The rest is the same as in Program 9-2, so do not type NEW. Restore Program 9-2 by reversing the changes you made to enter Program 9-3.

Program 9-4 Walking Bug with Blinking Eyes and Open Mouth

Do not type NEW.

```
100 REM PROGRAM 9-4
172 CALL HCHAR(6,C+4,64) ← Eye o
174 CALL HCHAR(7,C+5,61) ← Mouth =
202 CALL HCHAR(6,C+4,42) ← Eye *
204 CALL HCHAR(7,C+5,45) ← Mouth -
RUN
```

The first eye and mouth are in the bug-drawing subroutine. By changing pictures, we can make any figure walk. To have little aliens make faces as they move, we use the following three pictures:

Picture #1	Picture #2	Picture #3
\--/ (00) /--\	\--/ <*> /--\	\--/ {--} /--\

Since only the second line changes, we need four subroutines, one for the top and bottom lines and three for the middle lines. A fifth subroutine erases the entire picture. The middle lines consist of

	Drawing	Character codes
Picture #1	(00)	40 79 79 41
Picture #2	<*>	60 42 42 62
Picture #3	{--}	123 45 45 125

Program 9-5 Funny, Moving Aliens

```

NEW
100 REM PROGRAM 9-5
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW ALIEN'S TOP AND BOTTOM
130 GOSUB 1000
135 REM DRAW MIDDLE FOR PICTURE #1
140 GOSUB 2000
150 FOR K=1 TO 10
160 NEXT K
165 REM DRAW MIDDLE FOR PICTURE #2
170 GOSUB 3000
180 FOR K=1 TO 10
190 NEXT K
195 REM DRAW MIDDLE FOR PICTURE #3
200 GOSUB 4000
210 FOR K=1 TO 10
220 NEXT K
225 REM ERASE PICTURE
230 GOSUB 5000
240 NEXT C
250 GOTO 9000
995 REM TOP AND BOTTOM
999 REM TOP \--/
1000 CALL HCHAR(5,C,92)
1010 CALL HCHAR(5,C+1,45,2)
1020 CALL HCHAR(5,C+3,47)
1025 REM DRAW BOTTOM /--\
1030 CALL HCHAR(7,C,47)
1040 CALL HCHAR(7,C+1,45,2)
1050 CALL HCHAR(7,C+3,92)
1060 RETURN

```

```

1995 REM MIDDLE FOR PICTURE #1 (00)
2000 CALL HCHAR(6,C,40)
2010 CALL HCHAR(6,C+1,79,2)
2020 CALL HCHAR(6,C+3,41)
2030 RETURN
2995 REM MIDDLE FOR PICTURE #2 (***)
3000 CALL HCHAR(6,C,60)
3010 CALL HCHAR(6,C+1,32,2)
3020 CALL HCHAR(6,C+3,62)
3030 RETURN
3995 REM MIDDLE FOR PICTURE #3 (--)
4000 CALL HCHAR(6,C,123)
4010 CALL HCHAR(6,C+1,45,2)
4020 CALL HCHAR(6,C+3,125)
4030 RETURN
4995 REM ERASE PICTURE
5000 FOR R=5 TO 7
5010 CALL HCHAR(R,C,32,4)
5020 NEXT R
5030 RETURN
9000 CALL CLEAR
RUN

```

Programs 9-2 through 9-5 changed the appearances of objects as they move. If you want a creature just to make funny faces, simply keep its pictures in fixed positions. The pictures we will use are

Picture #1

```

|-----|
|  O  O  |
|   X   |
|  ==  |
|-----|

```

Picture #2

```

|-----|
|  @  @  |
|   O   |
|  VVV  |
|-----|

```

Picture #3

```

|-----|
|  X  X  |
|   @   |
|  ^^  |
|-----|

```

The outline is the same in all three pictures. Only the eyes, nose, and mouth change. The character codes for the changes are

Picture #1

```

O O      79      79
X         88
==       61 61 61

```

Picture #2

```

@ @      64      64
O         79
VVV      86 86 86

```

Picture #3

```

X X      88      88
*         42
^^       94 94 94

```

Program 9-6 Stationary Creature Making Faces

```

NEW
100 REM PROGRAM 9-6
110 CALL CLEAR
115 REM DRAW OUTLINE
120 GOSUB 1000
130 FOR J=1 TO 10
135 REM DRAW FACE #1
140 GOSUB 2000
145 REM DRAW FACE #2
150 GOSUB 3000
155 REM DRAW FACE #3
160 GOSUB 4000
170 NEXT J
180 GOTO 9000

```

Remember that REM statements are optional and can be omitted to save time in typing.

```

995 REM DRAW OUTLINE OF FACE
1000 CALL VCHAR(5,10,124,5)
1010 CALL VCHAR(5,16,124,5)
1020 CALL HCHAR(5,11,45,5)
1030 CALL HCHAR(9,11,45,5)
1040 RETURN
1995 REM DRAW FACE #1
2000 CALL HCHAR(6,12,79)
2010 CALL HCHAR(6,14,79)
2020 CALL HCHAR(7,13,88)
2030 CALL HCHAR(8,12,61,3)
2040 RETURN
2995 REM DRAW FACE #2
3000 CALL HCHAR(6,12,64)
3010 CALL HCHAR(6,14,64)
3020 CALL HCHAR(7,13,79)
3030 CALL HCHAR(8,12,86,3)
3040 RETURN
3995 REM DRAW FACE #3
4000 CALL HCHAR(6,12,88)
4010 CALL HCHAR(6,14,88)
4020 CALL HCHAR(7,13,42)
4030 CALL HCHAR(8,12,94,3)
4040 RETURN
9000 CALL CLEAR
RUN

```

Note that we have two VCHARs followed by two HCHARs.

Eyes O O

Nose X

Mouth ===

Eyes @ @

Nose O

Mouth VVV

Eyes X X

Nose @

Mouth ^^^

To slow the facial changes, insert

```

141 FOR K=1 TO 10
142 NEXT K

151 FOR K=1 TO 10
152 NEXT K

161 FOR K=1 TO 10
162 NEXT K

```

10

A UNIVERSE OF CHANGE

We can easily place objects randomly on the screen. BASIC has a special way to pick a random number, one that is equally likely to have any value within its limits. This is like rolling an honest die, which is equally likely to give any value from 1 to 6. On the TI-99/4A,

$K = \text{INT}(\text{RND} * N) + 1$

Note: * (not X) means "multiply" in BASIC.

makes the computer pick a random whole number K between 1 and N . RND selects a random number between 0 and 1, while INT makes $\text{RND} * N$ into a whole number (or *integer*). Note that RND is always less than 1, so $\text{RND} * N$ is always less than N . For example,

$D = \text{INT}(\text{RND} * 6) + 1$

produces a random number between 1 and 6 that could simulate the rolling of a die. Similarly,

$R = \text{INT}(\text{RND} * 38) + 1$

simulates a random spin of a roulette wheel (one color only). To select an arbitrary position on the screen, we need two random numbers. The row (line) number R must be between 1 and 24, while the column number C must be between 1 and 32.

To select a random row (1 to 24), we use

$R = \text{INT}(\text{RND} * 24) + 1$ ← Will be a number between 1 and 24

To select a random column (1 to 32), we use

```
C=INT(RND*32)+1
```

← Will be a number between 1 and 32

The following program places asterisks at random positions, creating the effect of falling snowflakes.

Program 10-1 Snowfall

```
NEW
100 REM PROGRAM 10-1
110 CALL CLEAR
120 FOR J=1 TO 30
130 R=INT(RND*24)+1
140 C=INT(RND*32)+1
150 CALL HCHAR(R,C,42)
160 NEXT J
170 CALL CLEAR
RUN
```

← We will draw 30 objects.

← The computer makes up a number between 1 and 24.

← The computer makes up a number between 1 and 32.

← Asterisk is character code 42.

The computer selects the vertical and horizontal positions of the asterisks randomly. To slow down the rate at which asterisks appear, insert

```
155 FOR K=1 TO 200
156 NEXT K
```

We can also make each asterisk disappear by adding

```
158 CALL HCHAR(R,C,32)
```

← Space is character code 32.

The result looks like a firefly flitting rapidly around the screen. You can slow it down by making the upper value in line 155 300 or 500 instead of 200. 500 makes the scene look like a picture taken by a slow-motion camera.

A more complex figure is the bug from Program 6-6. It occupies three lines and five columns.

```
000
-! !-
-!-!-
```

We must make sure the bug fits everywhere the computer is to print it.

Position 22, 1

```
000
-! !-
-!-!-
```

Position 24, 1

Position 22, 28

```
000
-! !-
-!-!-
```

Position 24, 32

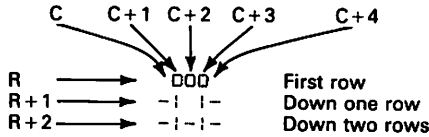
This drawing shows that the bug cannot fit if it starts beyond row 22 or column 28. Consequently, R and C cannot be larger than 22 and 28, respectively. So we must use

$R = \text{INT}(\text{RND} * 22) + 1$

and

$C = \text{INT}(\text{RND} * 28) + 1$

To draw the bug at random positions, we must use R and C throughout the picture.



Program 10-2 Bugs Appearing and Disappearing at Random Positions

```

NEW
100 REM PROGRAM 10-2
110 CALL CLEAR
120 FOR J=1 TO 20
130 R=INT(RND*22)+1
140 C=INT(RND*28)+1
145 REM DRAW THE BUG BEGINNING AT LOCATION R,C
150 CALL HCHAR(R,C+1,79,3)
160 CALL VCHAR(R+1,C,45,2)
170 CALL VCHAR(R+1,C+1,124,2)
180 CALL VCHAR(R+1,C+3,124,2)
190 CALL VCHAR(R+1,C+4,45,2)
200 CALL HCHAR(R+2,C+2,45)
205 REM SLOW COMPUTER DOWN
210 FOR K=1 TO 300
220 NEXT K
225 REM ERASE BUG
230 FOR ROW=R TO R+2
240 CALL HCHAR(ROW,C,32,5)
250 NEXT ROW
260 NEXT J
270 CALL CLEAR
RUN

```

← Draw 20 bugs.
 ← Eyes 000
 ← Left legs
 ← Left side of body
 ← Right side of body
 ← Right legs
 ← Bottom, another HCHAR

Note that lines 150 and 200 are the only HCHARs. If you want to slow the bug down, insert

```

235 FOR K=1 TO 200
236 NEXT K

```

If you want to make the bug stay in each place longer, change the 300 in line 210 to 500.

Program 10-2 draws the bug, starting at a random position, and then makes it disappear by printing three rows of spaces over it. Note how we got the R and C values for HCHAR and VCHAR from the picture of the bug. This paper drawing helped establish the values for each of the bug's parts. To draw a picture on the computer, we must

1. Draw the picture on paper.
2. Put the screen locations into the picture.
3. Write down the HCHAR and VCHAR statements.
4. Enter the program into the computer.

Writing programs or drawing pictures is easy if you take your time and plan the project ahead of time.

We can combine random positioning with the idea of multiple pictures to create an exploding star like the ones you often see in video games. Our three star pictures are

```

  O      ' '      \ /
         ' O      *
         ' '      / \
  
```

The character codes are

```

          96  39      92  47
79         48         42
          39  96      47  92
  
```

Since this picture in its extended form is three lines high and three columns wide, we must restrict its starting position to rows 1 through 22 and columns 1 through 30.

Program 10-3 Exploding Stars at Random Positions

```

NEW
100 REM PROGRAM 10-3
110 CALL CLEAR
120 FOR J=1 TO 20 ← Draw 20 stars.
130 R=INT(RND*22)+1 ← Random row, 1 to 22
140 C=INT(RND*30)+1 ← Random column, 1 to 30
145 REM DRAW ORIGINAL STAR CENTERED AT R+1,C+1
150 CALL HCHAR(R+1,C+1,79)
160 FOR K=1 TO 50
170 NEXT K
175 REM DRAW START OF EXPLOSION
180 CALL HCHAR(R,C,96)
190 CALL HCHAR(R,C+2,39)
200 CALL HCHAR(R+1,C+1,48)
210 CALL HCHAR(R+2,C,39)
220 CALL HCHAR(R+2,C+2,96)
230 FOR K=1 TO 50
240 NEXT K
245 REM DRAW FULL EXPLOSION
250 CALL HCHAR(R,C,92)
260 CALL HCHAR(R,C+2,47)
270 CALL HCHAR(R+1,C+1,42)
280 CALL HCHAR(R+2,C,47)
290 CALL HCHAR(R+2,C+2,92)
300 FOR K=1 TO 50
310 NEXT K
315 REM ERASE REMNANTS
320 FOR ROW=R TO R+2
330 CALL HCHAR(ROW,C,32,3)
340 NEXT ROW
350 NEXT J
RUN
  
```

To speed up or slow down the explosion, change the upper limits in the time-wasting loops in lines 160, 230, and 300.

We can print any keyboard character from Table 6-1. Let's redo Program 10-1, but instead of printing asterisks, let's print a different character each time. We can select a character randomly from Table 6-1 in any of the following ways:

1. A random capital letter. The random number must lie in the range 65 to 90 inclusive, so we need

```
CL=INT(RND*26)+65
```

2. A random digit. The random number must lie in the range 48 to 57 inclusive, so we need

```
D=INT(RND*10)+48
```

3. A random lowercase letter. The random number must lie in the range 97 to 122 inclusive, so we need

```
LL=INT(RND*26)+97
```

4. Any character. The random number must lie in the range 33 to 126 inclusive, so we need

```
CH=INT(RND*95)+33
```

Program 10-1 modified to scatter random characters is as follows:

Program 10-4 Random Characters at Random Positions

```
NEW
100 REM PROGRAM 10-4
110 CALL CLEAR
120 FOR J=1 TO 100
130 R=INT(RND*24)+1
140 C=INT(RND*32)+1
150 CH=INT(RND*95)+33
160 CALL HCHAR(R,C,CH) ← Puts random character CH at row R,
170 NEXT J               column C
180 CALL CLEAR
RUN
```

To leave the characters on the screen, delete line 180. The only problem this creates is that the message **** DONE **** appears at the bottom when the program ends, distorting the picture. If we change line 180 to

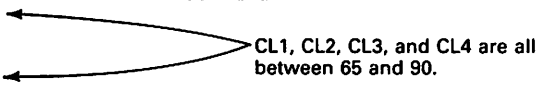
```
180 GOTO 120
```

the program will run forever, filling the screen with a living, constantly changing picture. To stop the revised program, press FCTN and 4 (CLEAR) together. To change the mixture of symbols, change the values in line 150.

We can use random letters to generate random four-letter words. How do you think they will turn out?

Program 10-5 Random Four-Letter-Word Generator

```
NEW
100 REM PROGRAM 10-5
110 CALL CLEAR
120 FOR J=1 TO 50
125 REM GENERATE FOUR RANDOM CAPITAL LETTERS
130 CL1=INT(RND*26)+65
140 CL2=INT(RND*26)+65
150 CL3=INT(RND*26)+65
160 CL4=INT(RND*26)+65
165 REM DISPLAY WORD NEAR CENTER OF SCREEN
170 CALL HCHAR(10,15,CL1)
180 CALL HCHAR(10,16,CL2)
190 CALL HCHAR(10,17,CL3)
200 CALL HCHAR(10,18,CL4)
210 FOR K=1 TO 200
220 NEXT K
230 NEXT J
240 CALL CLEAR
RUN
```



CL1, CL2, CL3, and CL4 are all between 65 and 90.

If you want a different sequence of random letters each time you run the program, insert

```
105 RANDOMIZE
```

Unfortunately (or fortunately, if you are a concerned teacher or parent), most four-letter combinations don't make much sense, although they could serve as government acronyms. Run Program 10-5 for a while and see how long it takes the computer to generate a real word. One way to improve your chances is to change line 140 to

```
140 CL2=2*INT(RND*11)+65
```

Examine Table 6-1 to see why this increases the likelihood of L2's being a vowel.

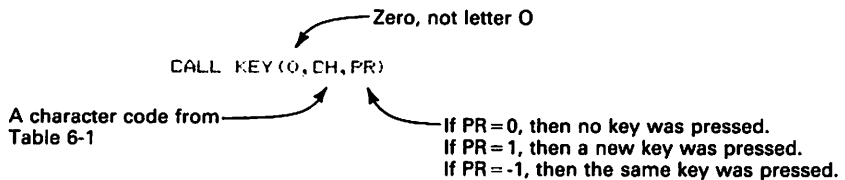
Note that lines 130 through 160 can produce four different letter codes although they all look the same. RND, remember, produces a different random number each time.

The reason the four-letter combinations are so disappointing is that letters in English words are not random. Some, such as vowels and the consonants *s*, *t*, *r*, *d*, and *n*, are very common, while others, such as *j*, *k*, *q*, and *z*, are uncommon. Furthermore, some pairs, such as *ch*, *sh*, and *th* are common, while others, such as *kq*, *lz*, *qd*, and *nj*, are either uncommon or nonexistent.

11

CONTROLLING ACTION FROM THE KEYBOARD

Program 8-9 drew a space station and had it fire a missile. All we did was type RUN; we could not control when the missile was fired. To control the firing from the keyboard (as in video games), we need the new BASIC statement CALL KEY. CALL KEY makes the computer stop and wait for the player to press a key. Its general form is



We use lines such as

```
300 CALL KEY (O, CH, PR)
310 IF PR=0 THEN 300
```

Do line 300 next if no key was pressed.

to wait for the player to press a key.

We can easily revise Program 8-9 to allow the player, not the computer, to decide when to fire a missile. We start with

```
NEW
110 CALL CLEAR
115 REM DRAW SPACE STATION
120 CALL VCHAR (22, 13, 124, 2)
130 CALL VCHAR (22, 19, 124, 2)
140 CALL HCHAR (22, 14, 45, 5)
150 CALL HCHAR (23, 14, 45, 5)
160 CALL HCHAR (22, 16, 94)
```

Be sure to press ENTER after each line.

Note the change from VCHAR to HCHAR.

```

165 REM FIRE MISSILE
170 FOR R=19 TO 1 STEP -1
180 CALL HCHAR(R,16,94) ← Draw top of missile ^ .
190 CALL HCHAR(R+1,16,124) ← Draw bottom of missile ! .
195 REM SLOW COMPUTER DOWN
200 FOR K=1 TO 50
210 NEXT K
215 REM ERASE MISSILE
220 CALL VCHAR(R,16,32,2) ← Prints 2 spaces vertically,
230 NEXT R                      erasing the missile.
240 CALL CLEAR

```

When you type RUN, the space station fires a missile automatically. To make it wait for you to tell it to fire, add the lines

```

165 REM WAIT FOR PLAYER TO PRESS ANY KEY
166 CALL KEY(0,CH,PR)
167 IF PR=0 THEN 166 ← Do 166 next if no key was
                        pressed.

```

Program 11-1 is the complete program with the addition, renumbered using RES.

Program 11-1 Space Station Firing a Missile under Keyboard Control

```

NEW
100 REM PROGRAM 11-1
110 CALL CLEAR
115 REM DRAW SPACE STATION
120 CALL VCHAR(22,13,124,2)
130 CALL VCHAR(22,19,124,2)
140 CALL HCHAR(22,14,45,5)
150 CALL HCHAR(23,14,45,5)
160 CALL HCHAR(22,16,94)
165 REM WAIT FOR PLAYER TO PRESS ANY KEY
170 CALL KEY(0,CH,PR)
180 IF PR=0 THEN 170 ← Do 170 next if no key was pressed.
190 FOR R=19 TO 1 STEP -1
200 CALL HCHAR(R,16,94) ← Draw top of missile ^ .
210 CALL HCHAR(R+1,16,124) ← Draw bottom of missile ! .
215 REM SLOW THE COMPUTER DOWN
220 FOR K=1 TO 50
230 NEXT K
235 REM ERASE MISSILE
240 CALL VCHAR(R,16,32,2) ← Prints 2 spaces vertically.
250 NEXT R
260 CALL CLEAR
RUN

```

After you type RUN, you must touch a key (say, F for *fire*) to make the station fire a missile. If you do not press a key, nothing happens. You now control the firing. Adding the following line lets the station fire repeatedly:

```
260 GOTO 170
```

Now the station fires a missile every time you press F. Of course, you must wait for the old missile to disappear, since the computer moves it all the way to the top before

examining the keyboard again. You must press FCTN and 4 (CLEAR) simultaneously to end the program.

Video game manufacturers use this kind of control in arcade games. Rockets, spaceships, or creatures move across the screen, and you must press a button (or a key) to fire at them. We will now draw moving targets and have our space station fire missiles at them. The targets are eye creatures that look like this:

<oo> Lowercase O(character code 111)
↖

Program 11-2 Eye Creatures Moving Right

```
NEW
100 REM PROGRAM 11-2
110 CALL CLEAR
115 REM DRAW CREATURES
120 FOR C=1 TO 25
130 CALL HCHAR(1,C,40)
140 CALL HCHAR(1,C+1,111,2)
150 CALL HCHAR(1,C+3,41)
155 REM SLOW COMPUTER DOWN
160 FOR K=1 TO 25
170 NEXT K
175 REM ERASE CREATURES
180 CALL HCHAR(1,C,32,4)
190 NEXT C
200 CALL CLEAR
RUN
```

The targets move right at the top of the screen. Let us now combine the moving targets with Program 11-1. We will make the computer look specifically for the F key (code 70) rather than just any key.

Program 11-3 Target Practice

```
NEW
100 REM PROGRAM 11-3
110 CALL CLEAR
115 REM DRAW SPACE STATION
120 GOSUB 1000
130 FOR C=1 TO 25
135 REM MOVE CREATURES RIGHT
140 GOSUB 2000
145 REM PRESS F KEY TO FIRE MISSILE
150 CALL KEY(0,CH,PR)
160 IF CH=70 THEN 220
165 REM ERASE CREATURES
170 FOR K=1 TO 25
180 NEXT K
190 CALL HCHAR(1,C,32,4)
200 NEXT C
210 GOTO 130
215 REM FIRE MISSILE
220 FOR R=19 TO 1 STEP -1
230 CALL HCHAR(R,16,94)
240 CALL HCHAR(R+1,16,124)
245 REM SLOW COMPUTER DOWN
250 FOR K=1 TO 50
```

F is character code 70.

Do 220 next if player pressed F.


```

260 NEXT K
265 REM ERASE MISSILE
270 CALL VCHAR(R,16,32,2) ← Prints 2 spaces vertically
280 NEXT R
290 GOTO 130
995 REM DRAW SPACE STATION ← Enter NUM 1000 before typing the
1000 CALL VCHAR(22,13,124,2)      subroutine.
1010 CALL VCHAR(22,19,124,2)
1020 CALL HCHAR(22,14,45,5)
1030 CALL HCHAR(23,14,45,5)
1040 CALL HCHAR(22,16,94)
1050 RETURN
1995 REM DRAW CREATURES ← Enter NUM 2000 before typing the
2000 CALL HCHAR(1,C,40)      subroutine.
2010 CALL HCHAR(1,C+1,111,2)
2020 CALL HCHAR(1,C+3,41)
2030 FOR K=1 TO 25
2040 NEXT K
2050 RETURN
RUN

```

Be sure to press ENTER after each line.

We place CALL KEY inside the moving creature instructions:

```

FOR C=1 TO 25 ←
GOSUB 2000 ←
NEXT C ←

```

Moves the creature

We also placed the lines that erase the eye creatures (165 through 200) below CALL KEY to make the creature visible when the missile is launched.

If the player does not press F, the computer keeps moving the eye creatures. If the player presses F, the computer goes to line 220 and fires a missile. The line

```
CALL KEY(0,CH,PR)
```

checks the keyboard without stopping the program. Note, however, that as soon as you press F, the creatures stop because the computer has left the part of the program that moved them.

To speed the creatures up and make them more difficult to hit, change line 2030 to

```
2030 FOR K=1 TO 10
```

Note that lines 210 and 290 both restart the creatures at the left edge of the screen. Don't forget to press F to fire a missile. Hold F down until the missile appears; a quick touch will often have no effect.

Be careful when typing lines 1000, 1010, and 2010. They all extend exactly to the right-hand boundary of the screen. When you finish one of them, the cursor will move down to the next screen line, anticipating more typing. You might think you do not have to press ENTER, since the cursor has already moved down. However, if you start typing the next program line without pressing ENTER, the computer will think you are continuing the current program line; the result will be an error message. So press ENTER at the end of each program line.

Note that you should see a blank line on the screen after lines 1000, 1010, and 2010. For example, lines 1000 through 1030 should appear as follows:

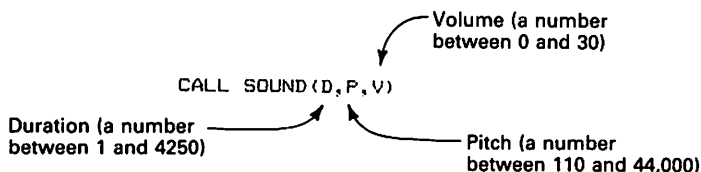
```
1000 CALL VCHAR(22,13,124,2)
1010 CALL VCHAR(22,19,124,2)
1020 CALL HCHAR(22,14,45,5)
1030 CALL HCHAR(23,14,45,5)
```

Another way to remind yourself to press ENTER is to look for the prompt character (>). You cannot start a new program line unless that character appears at the far left. It will not appear on screen lines that are simply continuations of previous lines.

12

MAKING NOISE

Arcade games play music and make noise. We can do the same things with the computer. To make noise, we use the BASIC statement



The duration (D) is how long the sound lasts; the value 1000 lasts a second. A low pitch (P) value produces low notes, and a high value produces high notes (values over 20,000 are difficult to hear). Be sure to keep P between 110 and 44000. A volume (V) of 0 is very loud, whereas 30 is the quietest; note that higher V values result in lower volumes.

Let us vary the pitch in steps of 50 units, starting at 200.

Program 12-1 Simple Sound

(Be sure to turn up the volume on your television set.)

```
100 REM PROGRAM 12-1
110 CALL CLEAR
120 FOR P=200 TO 2000 STEP 50
130 CALL SOUND(500,P,15)
140 NEXT P
RUN
```

← P starts at 200 and goes in steps of 50 (250, 300, 350) to 2000.

Each sound produced lasts $\frac{1}{2}$ second and is halfway up the volume scale (15 is halfway between 0 and 30). Note how the sound changes from a pleasant melody to a high-pitched whine that hurts the ears. To get even higher pitches, change line 120 to

```
120 FOR P=2000 TO 4000 STEP 50
```

To see the value of the pitch (P) in each iteration, add the line

```
125 PRINT P
```

By using random numbers, we can produce sounds with random pitches. This will sound like modern atonal music.

Program 12-2 Random Sounds

```
NEW
100 REM PROGRAM 12-2
110 CALL CLEAR
120 FOR J=1 TO 20
130 P=INT(RND*1000)+110
140 CALL SOUND(100,P,15)
150 NEXT J
RUN
```

← We want 20 sounds.

← Computer selects a number randomly between 110 and 1109.

We can combine several sounds to compose a little computer song.

Program 12-3 Computer Song

```
NEW
100 REM PROGRAM 12-3
110 CALL CLEAR
120 FOR J=1 TO 10
130 P=INT(RND*1000)+110
140 CALL SOUND(200,P,15)
150 CALL SOUND(200,P+20,15)
160 CALL SOUND(100,P+30,15)
170 CALL SOUND(100,P+40,15)
180 CALL SOUND(100,P+50,15)
190 NEXT J
RUN
```

Our song, although not a classic, shows us that the computer can play music. By using a series of low notes and high notes, we can make sounds to accompany a variety of pictures. We can, for example, create blastoff noises to go with rocket launches, strange sounds to accompany alien creatures, and shrieking sirens for police cars. Let us first produce sound for a rocket launch. Be sure to turn up the volume on your television set.

Program 12-4 Rocket Launch with Sound

```
NEW
100 REM PROGRAM 12-4
110 CALL CLEAR
115 REM DRAW ROCKET
120 PRINT "  /\  "
130 PRINT " /  \ "
140 PRINT "/    \ "
150 PRINT " !MARS!"
160 PRINT " !SHIP!"
170 PRINT "  !  ! "
180 PRINT "  !  ! "
190 PRINT "  ----  "
200 PRINT " /\  /\  "
210 PRINT "  ----  "
215 REM MOVE ROCKET UP, PUT SMOKE UNDERNEATH
220 FOR J=1 TO 24
230 PRINT " **** "
235 REM PRODUCE WEIRD NOISE
240 CALL SOUND(10,2000,20)
250 CALL SOUND(10,130,20)
260 CALL SOUND(20,2000,20)
270 CALL SOUND(20,110,20)
280 NEXT J
290 CALL CLEAR
RUN
```

3 spaces

By changing from a high pitch of 2000 to low pitches of 130 and 110, we produce a weird sound to accompany the rocket. In fact, this sound might be better suited to announce the arrival of the cavalry in a Western movie. We can also produce plain (or *white*) noise by using a negative value such as -2 or -3 for P in CALL SOUND. To change Program 12-4 to make white noise, add the following lines.

Program 12-5 Rocket Launch with Noise

Do not type NEW. Leave Program 12-4 in the computer.

```
100 REM PROGRAM 12-5
240 CALL SOUND(10,-2,15)
250 CALL SOUND(10,-3,15)
260 CALL SOUND(20,-2,15)
270 CALL SOUND(20,-3,15)
RUN
```

Negative P values produce noise.

Let's write a program that makes strange eyes look at us and produce weird noises. We use the eye creatures from Program 7-5.

Program 12-6 Eye Creatures Making Noises

```
NEW
100 REM PROGRAM 12-6
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW EYES (OO)
130 CALL HCHAR(S,C,40)
140 CALL HCHAR(S,C+1,79,2)
150 CALL HCHAR(S,C+3,41)
155 REM PRODUCE SOUND
160 FOR K=1 TO 3
170 CALL SOUND(100,130,10)
180 CALL SOUND(200,500,10)
190 NEXT K
195 REM ERASE PICTURE
200 CALL HCHAR(S,C,32,4)
210 NEXT C
220 CALL CLEAR
RUN
```

To make the eye creatures blink while they make noises, we must draw another picture with closed eyelids.

Program 12-7 Blinking Eye Creatures Making Noise

Do not type NEW. Leave Program 12-6 in the computer.

```
NEW
100 REM PROGRAM 12-7
195 REM DRAW BLINKING EYES (--)
200 CALL HCHAR(S,C+1,45,2)
205 REM PRODUCE SOUND FOR BLINK
210 FOR K=1 TO 3
220 CALL SOUND(100,200,15)
230 CALL SOUND(150,150,15)
240 NEXT K
245 REM ERASE PICTURE
250 CALL HCHAR(S,C,32,4)
260 NEXT C
270 CALL CLEAR
RUN
```

We do not need to slow Program 12-7 down because the computer slows down automatically when making sound. It takes time to send sound signals to the speakers inside the television set. The longer the sound lasts, the slower the picture will change. If the movement is too slow or too fast, you can

1. Change the upper value in lines 160 and 210 to produce more or less sound for each picture.
2. Increase or decrease the duration in each CALL SOUND statement.
3. Insert the lines

```

153 FOR K=1 TO 50
154 NEXT K

203 FOR K=1 TO 50
204 NEXT K

```

Another slowdown method uses CALL SOUND.

Program 12-8 Eye Creatures Moving Slowly

```

NEW
100 REM PROGRAM 12-8
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW THE EYES (OO)
130 CALL HCHAR(S,C,40)
140 CALL HCHAR(S,C+1,79,2)
150 CALL HCHAR(S,C+3,41)
155 REM SLOW EYES DOWN
160 CALL SOUND(500,30000,30)
165 REM DRAW BLINKING EYES (--)
170 CALL HCHAR(S,C+1,45,2)
175 REM PRODUCE SOUND FOR BLINK
180 CALL SOUND(500,30000,30)
185 REM ERASE PICTURE
190 CALL HCHAR(S,C,32,4)
200 NEXT C
210 CALL CLEAR
RUN

```

Remember the REM lines are optional and can be omitted to save typing.

Be sure to press ENTER after typing these lines.
Slows computer down

The computer moves the eye creatures right rather slowly because of the time taken by the SOUND commands. But what happened to the sound? The trick is that 30000 is too high-pitched for the human ear, so you don't hear anything. (If your dog begins to howl, try a higher value). The statement

```
CALL SOUND(500,30000,30)
```

lasts ½ second and has a pitch of 30000 and a volume of 30 (very quiet). To slow the action even more, change the duration to 1000 or 3000. Let us draw an alien creature and produce a weird sound. The alien looks like this:

```

\--/
[OO]
/--\

```

Program 12-9 Moving Alien Making Sound

```
NEW
100 REM PROGRAM 12-9
110 CALL CLEAR
120 FOR C=1 TO 20
125 REM DRAW ALIEN
130 GOSUB 1000
135 REM PRODUCE SOUNDS
140 GOSUB 2000
145 REM ERASE ALIEN
150 GOSUB 3000
160 NEXT C
170 GOTO 9000
995 REM DRAW ALIEN
999 REM DRAW TOP  \--/
1000 CALL HCHAR(5,C,92)
1010 CALL HCHAR(5,C+1,45,2)
1020 CALL HCHAR(5,C+3,47)
1025 REM DRAW MIDDLE [00]
1030 CALL HCHAR(6,C,91)
1040 CALL HCHAR(6,C+1,79,2)
1050 CALL HCHAR(6,C+3,93)
1055 REM DRAW BOTTOM /--\
1060 CALL HCHAR(7,C,47)
1070 CALL HCHAR(7,C+1,45,2)
1080 CALL HCHAR(7,C+3,92)
1090 RETURN
1995 REM PRODUCE SOUND
2000 CALL SOUND(20,135,10)
2010 CALL SOUND(10,155,10)
2020 CALL SOUND(10,300,20)
2030 CALL SOUND(20,220,10)
2040 RETURN
2995 REM ERASE ALIEN
3000 FOR R =5 TO 7
3010 CALL HCHAR(R,C,32,4)
3020 NEXT R
3030 RETURN
9000 CALL CLEAR
RUN
```

To make the alien turn starry-eyed as it moves, insert

```
1085 CALL HCHAR(6,C+1,42,2)
```


13

FALLING OBJECTS

We have moved things right, left, and up. Now we will move things down. The simple object we start with is

<*>

The following FOR-NEXT loop moves it down from row 1 to row 24:

```
FOR R=1 TO 24
CALL HCHAR(R, 16, 60) ← The <
CALL HCHAR(R, 17, 42) ← The *
CALL HCHAR(R, 18, 62) ← The >

NEXT R
```

The computer must also print spaces over the previous picture before drawing the current one. We use

```
CALL HCHAR(R, 16, 32, 3)
```

Character code for a space



to erase the object.

Let us first see what happens without the erasure.

Program 13-1 Falling Asterisk Without Erasure

```
NEW
100 REM PROGRAM 13-1
110 CALL CLEAR
120 FOR R=1 TO 24
125 REM DRAW OBJECT  <*>
130 CALL HCHAR(R,16,60)
140 CALL HCHAR(R,17,42)
150 CALL HCHAR(R,18,62)
160 NEXT R
170 CALL CLEAR
RUN
```

A whole column of objects appears in an instant! Inserting either

```
152 FOR K=1 TO 100
154 NEXT K
```

or

```
152 CALL SOUND(200,30000,30)
```

will slow the computer down. To avoid multiple images, the computer must print spaces over each picture.

Program 13-2 Falling Asterisk with Erasure

```
NEW
100 REM PROGRAM 13-2
110 CALL CLEAR
120 FOR R=1 TO 24
125 REM DRAW OBJECT  <*>
130 CALL HCHAR(R,16,60)
140 CALL HCHAR(R,17,42)
150 CALL HCHAR(R,18,62)
160 CALL SOUND(300,30000,30) ← Sound is too high-pitched for
170 CALL HCHAR(R,16,32,3)      you to hear.
180 NEXT R
190 CALL CLEAR
RUN
```

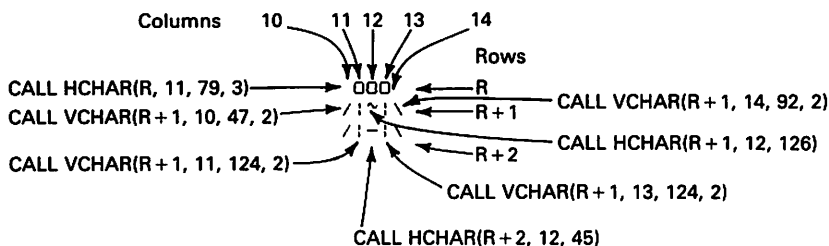
The asterisk floats down the screen like a descending parachutist. We can use a similar approach to make a more complex figure fall. Let us try a spider, which looks like this:

```
000
/!~!\
/!-!\
```

The character codes for the spider are

Picture	Character Codes
000	79 79 79
/!~!\	47 124 126 124 92
/!-!\	47 124 45 124 92

To begin the drawing in column 10, the screen locations and the corresponding HCHAR and VCHAR statements are



Program 13-3 Falling Spider

```

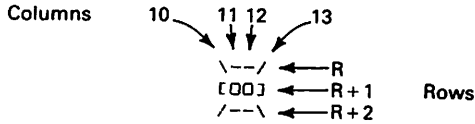
NEW
100 REM PROGRAM 13-3
110 CALL CLEAR
120 FOR R=1 TO 22
125 REM DRAW SPIDER
130 GOSUB 1000
135 REM PRODUCE SOUND
140 GOSUB 2000
145 REM ERASE SPIDER
150 GOSUB 3000
160 NEXT R
170 GOTO 9000
995 REM DRAW SPIDER
1000 CALL HCHAR(R, 11, 79, 3) ← Eyes 000 note the HCHAR.
1010 CALL VCHAR(R+1, 10, 47, 2) ← 2 left legs
1020 CALL VCHAR(R+1, 11, 124, 2) ← Left side of body
1030 CALL VCHAR(R+1, 13, 124, 2) ← Right side of body
1040 CALL VCHAR(R+1, 14, 92, 2) ← 2 right legs
1050 CALL HCHAR(R+1, 12, 126) ← Mouth
1060 CALL HCHAR(R+2, 12, 45) ← Bottom
1070 RETURN
1995 REM PRODUCE NOISE
2000 CALL SOUND(50, -3, 10)
2010 CALL SOUND(100, -2, 15)
2020 CALL SOUND(100, -3, 15)
2030 CALL SOUND(200, -2, 5) ← Produce white noise.
2040 RETURN
2995 REM ERASE SPIDER
3000 FOR ROW=R TO R+2
3010 CALL HCHAR(ROW, 10, 32, 5) ← Draw 3 rows of 5 spaces.
3020 NEXT ROW
3030 RETURN
9000 CALL CLEAR
RUN

```

Loud little fellow, isn't he? Good thing spiders don't make that much noise around the house! We had to change FOR R=1 TO 24 in Program 13-2 to FOR R=1 TO 22. The reason is that the spider occupies three lines, whereas the asterisk occupied only one. Note that the spider takes up lines R, R+1, and R+2. Remember that we must keep the picture on the 24-line screen. If R and C in CALL HCHAR and CALL VCHAR are not within their limits (1 to 24 for R and 1 to 32 for C), you will get an error and your program will not work. Using FOR R=1 TO 22 makes the largest row number 24 in

```
CALL HCHAR(R+2,12,45)
```

Let's make an alien creature fall. The picture is



Program 12-9 moved the alien right to left with sound. We use the same idea but now vary R (row number) instead of C (column number). We let the alien move down in column 10.

Program 13-4 Falling Alien

```
NEW
100 REM PROGRAM 13-4
110 CALL CLEAR
120 FOR R=1 TO 22
125 REM DRAW ALIEN
130 GOSUB 1000
135 REM PRODUCE SOUNDS
140 GOSUB 2000
145 REM ERASE ALIEN
150 GOSUB 3000
160 NEXT R
170 GOTO 9000
995 REM DRAW ALIEN
999 REM DRAW TOP      \--/
1000 CALL HCHAR(R,10,92)
1010 CALL HCHAR(R,11,45,2)
1020 CALL HCHAR(R,13,47)
1025 REM DRAW MIDDLE [00]
1030 CALL HCHAR(R+1,10,91)
1040 CALL HCHAR(R+1,11,79,2)
1050 CALL HCHAR(R+1,13,93)
1055 REM DRAW BOTTOM  /--\
1060 CALL HCHAR(R+2,10,47)
1070 CALL HCHAR(R+2,11,45,2)
1080 CALL HCHAR(R+2,13,92)
1090 RETURN
1995 REM PRODUCE SOUND
2000 CALL SOUND(20,135,10)
2010 CALL SOUND(10,155,10)
2020 CALL SOUND(10,300,20)
2030 CALL SOUND(20,220,10)
2040 RETURN
2995 REM ERASE ALIEN PICTURE
3000 FOR ROW=R TO R+2
3010 CALL HCHAR(ROW,10,32,4)
3020 NEXT ROW
3030 RETURN
9000 CALL CLEAR
RUN
```

To make the alien's eyes change shape as it falls, we change only the eye characters. Let's also add a second sound.

Program 13-5 Blinking, Falling Alien

Do not type NEW. Leave Program 13-4 in memory.

```
100 REM PROGRAM 13-5
145 REM DRAW ALIEN'S EYES STARRY
150 CALL HCHAR(R+1,11,42,2)
155 REM PRODUCE DIFFERENT SOUND
160 GOSUB 4000
165 REM ERASE ALIEN
170 GOSUB 3000
180 NEXT R
190 GOTO 9000
3995 REM PRODUCE DIFFERENT SOUND
4000 CALL SOUND(5,1000,15)
4010 CALL SOUND(5,300,15)
4020 CALL SOUND(2,200,15)
4030 CALL SOUND(5,1000,15)
4040 RETURN
RUN
```

Feel free to change the alien's eyes. We could use a series of eye drawings and slow the changes with a different sound for each picture. Use the following characters to produce more changes.

Eyes	Character Code
@@	64
--	46
^^	126
::	58

Program 13-6 Falling Alien Blinking Madly

```
NEW
100 REM PROGRAM 13-6
110 CALL CLEAR
120 FOR R=1 TO 22
125 REM DRAW ALIEN
130 GOSUB 1000
135 REM PRODUCE SOUNDS
140 GOSUB 2000
145 REM DRAW MULTIPLE EYES
150 GOSUB 4000
155 REM ERASE ALIEN
160 GOSUB 3000
170 NEXT R
180 GOTO 9000
995 REM DRAW ALIEN
999 REM DRAW TOP \--/
1000 CALL HCHAR(R,10,92)
1010 CALL HCHAR(R,11,45,2)
1020 CALL HCHAR(R,13,47)
1025 REM DRAW MIDDLE [00]
1030 CALL HCHAR(R+1,10,91)
1040 CALL HCHAR(R+1,11,79,2)
```

```

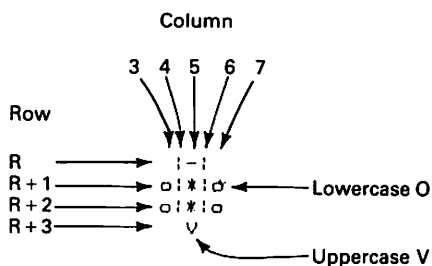
1050 CALL HCHAR(R+1,13,93)
1055 REM DRAW BOTTOM /--\
1060 CALL HCHAR(R+2,10,47)
1070 CALL HCHAR(R+2,11,45,2)
1080 CALL HCHAR(R+2,13,92)
1090 RETURN
1995 REM PRODUCE SOUND
2000 CALL SOUND(20,135,10)
2010 CALL SOUND(10,155,10)
2020 CALL SOUND(10,300,20)
2030 CALL SOUND(20,220,10)
2040 RETURN
2995 REM ERASE ALIEN PICTURE
3000 CALL HCHAR(R,10,32,4)
3010 CALL HCHAR(R+1,10,32,4)
3020 CALL HCHAR(R+2,10,32,4)
3030 RETURN

3995 REM DRAW MULTIPLE EYES
3999 REM DRAW @@ FOR EYES
4000 CALL HCHAR(R+1,11,64,2)
4010 CALL SOUND(30,500,15)
4015 REM DRAW .. FOR EYES
4020 CALL HCHAR(R+1,11,46,2)
4030 CALL SOUND(30,120,15)
4035 REM DRAW ^^ FOR EYES
4040 CALL HCHAR(R+1,11,126,2)
4050 CALL SOUND(30,1000,15)
4055 REM DRAW :: FOR EYES
4060 CALL HCHAR(R+1,11,58,2)
4070 CALL SOUND(30,225,15)
4080 RETURN
9000 CALL CLEAR
RUN

```

When you RUN Program 13-6, the alien falls, blinking furiously and making an undulating sound.

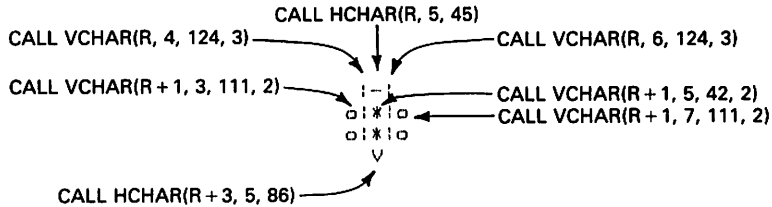
Let's now draw a race car moving down the screen. The car drawing is



Converting the picture into character codes produces

Picture	Character Codes
-	124 45 124
o!*!o	111 124 42 124 111
o!*!o	111 124 42 124 111
v	B6

The car begins at the top of the screen in column 3. The HCHAR and VCHAR commands to draw it are



Program 13-7 Race Car Moving Down

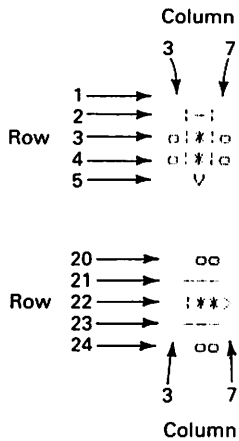
```

NEW
100 REM PROGRAM 13-7
110 CALL CLEAR
120 FOR R=1 TO 21
125 REM DRAW CAR
130 GOSUB 1000
135 REM PRODUCE SOUND
140 GOSUB 2000
145 REM ERASE CAR
150 GOSUB 3000
160 NEXT R
170 GOTO 9000
995 REM DRAW CAR FACING DOWN
1000 CALL VCHAR(R+1,3,111,2) ← Left wheels
1010 CALL VCHAR(R,4,124,3) ← Left frame
1020 CALL HCHAR(R,5,45) ← Rear. Note the HCHAR here.
1030 CALL VCHAR(R,6,124,3) ← Right frame
1040 CALL VCHAR(R+1,7,111,2) ← Right wheels
1050 CALL VCHAR(R+1,5,42,2) ← Center
1060 CALL HCHAR(R+3,5,86) ← Front. Note the HCHAR here.
1070 RETURN
1995 REM PRODUCE CAR SOUND
2000 CALL SOUND(100,-2,10)
2010 CALL SOUND(50,-3,5)
2020 CALL SOUND(10,-2,15)
2030 CALL SOUND(10,-3,2)
2040 RETURN
2995 REM ERASE CAR
3000 CALL HCHAR(R,3,32,5) ←
3010 CALL HCHAR(R+1,3,32,5) ←
3020 CALL HCHAR(R+2,3,32,5) ←
3030 CALL HCHAR(R+3,3,32,5) ← Draw 4 rows of 5 spaces.
3040 RETURN
9000 CALL CLEAR
RUN

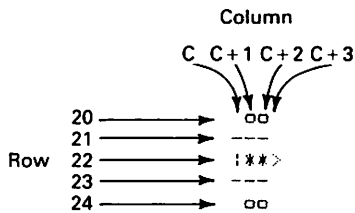
```

To make the car move down and then right, we need a separate car picture for each direction.

The row and column numbers indicate the position of the car on the race track.



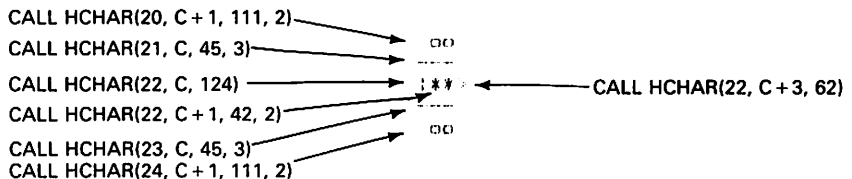
The car moving right looks like this:



Converting the car into character codes gives

Picture	Character Codes
oo	111 111
---	45 45 45
! ** >	124 42 42 62
---	45 45 45
oo	111 111

The car is four lines (rows) tall and begins in the lower left-hand corner on row 20. Remember, we cannot let R in HCHAR or VCHAR exceed 24. The HCHAR and VCHAR commands become



Program 13-8 Race Car Moving Down, Then Right

```
NEW
100 REM PROGRAM 13-8
110 CALL CLEAR
115 REM CAR MOVING DOWN
120 FOR R=1 TO 20
130 GOSUB 1000
135 REM PRODUCE SOUND
140 GOSUB 5000
145 REM ERASE CAR
150 GOSUB 1500
160 NEXT R
165 REM CAR MOVING RIGHT
170 FOR C=4 TO 28
180 GOSUB 2000
185 REM PRODUCE SOUND
190 GOSUB 5000
195 REM ERASE CAR
200 GOSUB 2500
210 NEXT C
220 GOTO 9000

995 REM CAR MOVING DOWN
1000 CALL VCHAR(R+1,3,111,2) ← Left wheels
1010 CALL VCHAR(R,4,124,3) ← Left frame
1020 CALL HCHAR(R,5,45) ← Rear
1030 CALL VCHAR(R,6,124,3) ← Right frame
1040 CALL VCHAR(R+1,7,111,2) ← Right wheels
1050 CALL VCHAR(R+1,5,42,2) ← Center
1060 CALL HCHAR(R+3,5,86) ← Front
1070 RETURN
1495 REM ERASE CAR MOVING DOWN
1500 FOR ROW=R TO R+3
1510 CALL HCHAR(ROW,3,32,5) ← Draw 4 rows of 5 spaces.
1520 NEXT ROW
1530 RETURN
1995 REM CAR MOVING RIGHT
2000 CALL HCHAR(20,C+1,111,2)
2010 CALL HCHAR(21,C,45,3)
2020 CALL HCHAR(22,C,124)
2030 CALL HCHAR(22,C+1,42,2)
2040 CALL HCHAR(22,C+3,62)
2050 CALL HCHAR(23,C,45,3)
2060 CALL HCHAR(24,C+1,111,2)
2070 RETURN
2495 REM ERASE CAR MOVING RIGHT
2500 FOR ROW=20 TO 24
2510 CALL HCHAR(ROW,C,32,4) ← Draw 5 rows of 4 spaces.
2520 NEXT ROW
2530 RETURN
4995 REM PRODUCE CAR SOUND
5000 CALL SOUND(2,-2,10)
5010 CALL SOUND(2,-3,5)
5020 CALL SOUND(2,-2,15)
5030 CALL SOUND(2,-3,2)
5040 RETURN
9000 CALL CLEAR
RUN
```

To make an alien spaceship fall, use the following drawing:

```
<=^=>
(000)
<=v=>
```



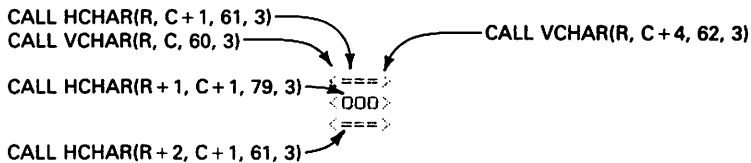
Lowercase V

The drawing contains 15 characters and could require 15 HCHAR and VCHAR statements. However, we can draw the basic picture as

```
<===>
(000)
<===>
```

and then add ^, v, (, and), much like a landscape artist who does a picture's background first and then paints the foreground objects on top of it. The HCHAR and VCHAR statements for the basic picture starting in row R and column C are

```
CALL HCHAR(R, C+1, 61, 3)
CALL VCHAR(R, C, 60, 3)
CALL HCHAR(R+1, C+1, 79, 3)
CALL HCHAR(R+2, C+1, 61, 3)
```



To print ^, v, (, and), we add the following statements:

```
CALL HCHAR(R, C+2, 94) -> ^
CALL HCHAR(R+2, C+2, 118) -> v
CALL HCHAR(R+1, C, 40) -> (
CALL HCHAR(R+1, C+4, 41) -> )
```

Program 13-9 Falling Spaceship

```
NEW
100 REM PROGRAM 13-9
110 CALL CLEAR
120 C=6
130 FOR R=1 TO 22
135 REM DRAW SPACESHIP
140 GOSUB 1000
145 REM PRODUCE SOUND
150 GOSUB 2000
155 REM ERASE SPACESHIP
160 GOSUB 3000
170 NEXT R
180 GOTO 9000
990 REM DRAW SPACESHIP
1000 CALL VCHAR(R,C,60,3)
1010 CALL HCHAR(R+1,C,40)
1020 CALL VCHAR(R,C+4,62,3)
```

Watch the changes from VCHAR to HCHAR.

```

1030 CALL HCHAR(R+1,C+4,41)
1040 CALL HCHAR(R,C+1,61,3)
1050 CALL HCHAR(R,C+2,94)
1060 CALL HCHAR(R+1,C+1,79,3)
1070 CALL HCHAR(R+2,C+1,61,3)
1080 CALL HCHAR(R+2,C+2,118)
1090 RETURN
1990 REM PRODUCE SOUND
2000 CALL SOUND(20,-2,10)
2010 CALL SOUND(20,130,10)
2020 CALL SOUND(10,1000,10)
2030 RETURN
2990 REM ERASE SPACESHIP
3000 FOR ROW=R TO R+2
3010 CALL HCHAR(ROW,C,32,5)
3020 NEXT ROW
3030 RETURN
9000 CALL CLEAR
RUN

```

The alien spaceship starts in column 6. To draw it starting in a random column, we can use

```
C=INT(RND*32)+1
```

The computer will give C a random value between 1 and 32 (the width of the screen for using graphics). The problem here is that the spaceship is five columns wide, and we do not want it to fall off the screen and produce an error message. So we must change the limit in C from 32 to 28.

Program 13-10 Falling Spaceship in Random Columns

Do not type NEW. Leave Program 13-9 in the computer.

```

100 REM PROGRAM 13-10
120 C=INT(RND*28)+1
180 GOTO 110
RUN

```

Remember, the computer will now drop spaceships forever. To stop it, press FCTN and 4 (CLEAR).

To make the alien spaceship move up and down, we can combine Program 13-10 with Program 8-9 (using a spaceship instead of a rocket). Since the spaceship looks the same regardless of its direction, we can use the same picture throughout.

***Program 13-11 Allen Spaceship Moving Up and Down
in Random Columns***

Do not type NEW. Leave Program 13-10 in the computer.

```
100 REM PROGRAM 13-11
175 REM MOVE SPACESHIP UP
180 FOR R=22 TO 1 STEP -1
185 REM DRAW SPACESHIP
190 GOSUB 1000
195 REM PRODUCE SOUND
200 GOSUB 2000
205 REM ERASE SPACESHIP
210 GOSUB 3000
220 NEXT R
230 GOTO 110
RUN
```

Because this program runs forever, you must use FCTN and 4 (CLEAR) to stop it.

14

BOMBING GAME

We are now ready to create a simple game in which the player tries to drop bombs on targets (peacefully, of course). Let us start with a falling bomb. With its HCHAR and VCHAR statements, it looks like this:

```
CALL HCHAR(1, 5, 95, 2) → --  
CALL HCHAR(2, 5, 124, 2) → !!  
CALL HCHAR(3, 5, 92) → \/  
                        ↖ CALL HCHAR(3, 6, 47)
```

Program 14-1 Falling Bomb

```
NEW  
100 REM PROGRAM 14-1  
110 CALL CLEAR  
120 FOR R=1 TO 22  
125 REM DRAW BOMB  
130 GOSUB 1000  
135 REM PRODUCE NOISE  
140 CALL SOUND(300,-2,20)  
145 REM ERASE BOMB  
150 GOSUB 2000  
160 NEXT R  
170 GOTO 9000  
990 REM DRAW BOMB  
1000 CALL HCHAR(R,5,95,2)  
1010 CALL HCHAR(R+1,5,124,2)  
1020 CALL HCHAR(R+2,5,92)  
1030 CALL HCHAR(R+2,6,47)  
1040 RETURN  
1990 REM ERASE BOMB  
2000 FOR ROW=R TO R+2  
2010 CALL HCHAR(ROW,5,32,2)  
2020 NEXT ROW  
2030 RETURN  
9000 CALL CLEAR  
RUN
```

We can easily draw ground at the bottom for the bomb to strike.

```
CALL HCHAR(24,1,95,32)
```

prints 32 underscore symbols (—) on the 24th row. To add targets, we can put Xs at every fifth position on the ground. The scene looks like this:

```
      --  
      ||  
      \/
```



```
      X   X   X   X   X   X  
      |   |   |   |   |  
Column 5  10  15  20  25  30
```

The following statements produce the ground and targets:

```
114 REM DRAW GROUND  
115 CALL HCHAR(24,1,95,32)  
116 FOR C=5 TO 30 STEP 5  
117 CALL HCHAR(24,C,88)  
118 NEXT C
```

Stepping C by 5 (5, 10, 15, 20, 25, 30) places the X targets in the desired columns.

Program 14-2 Bomb Falling to the Ground, with Noise

Do not type NEW. Leave Program 14-1 in memory.

```
100 REM PROGRAM 14-2  
114 REM DRAW GROUND  
115 CALL HCHAR(24,1,95,32)  
116 FOR C=5 TO 30 STEP 5  
117 CALL HCHAR(24,C,88)  
118 NEXT C  
RUN
```

Enter NUM 114, 1 before typing
the additions.

We can use a similar approach to drop a small bomb that looks like this:

(*)

Let us make it begin in column 16.

Program 14-3 Small Bomb Falling, with Noise

```
NEW
100 REM PROGRAM 14-3
110 CALL CLEAR
115 REM DRAW GROUND WITH TARGETS
120 GOSUB 2000
125 REM PLACE BOMB IN COLUMN 16
130 C=16
140 FOR R=1 TO 24
145 REM DRAW SMALL BOMB
150 GOSUB 1000
155 REM PRODUCE NOISE
160 CALL SOUND(300,-2,20)
165 REM ERASE BOMB
170 CALL HCHAR(R,C,32,3)
180 NEXT R
190 GOTO 9000
990 REM DRAW SMALL BOMB (*)
1000 CALL HCHAR(R,C,40)
1010 CALL HCHAR(R,C+1,42)
1020 CALL HCHAR(R,C+2,41)
1030 RETURN
1990 REM DRAW GROUND WITH TARGETS
2000 CALL HCHAR(24,1,95,32)
2010 FOR C=5 TO 30 STEP 5
2020 CALL HCHAR(24,C,88)
2030 NEXT C
2040 RETURN
9000 END
RUN
```

A bomb falls in column 16 when we type RUN. Note how it breaks through the ground and targets, leaving a sizable gap.

The END in line 9000 simply tells the computer to stop the program. We do not use CALL CLEAR as in other programs because we want to see the gruesome results of the bombing raid.

To let the player control when the bomb falls, we use CALL KEY, as in Program 11-1. We must add the statements

```
134 REM PRESS F KEY TO DROP BOMB
135 CALL KEY(<O,CH,PR>)
136 IF CH<>70 THEN 135
```

Zero, not letter O

This freezes the computer until the player presses the F key. The sequence <> (that is, *less than or greater than*) means “not equal to.” To type it, press the < key (shifted comma) first, then the > key (shifted period). We must also add

```
132 R=1
133 GOSUB 1000
```

to draw the bomb at the top initially.

The steps in dropping the bomb are

1. Clear the screen.
2. Draw a picture of the bomb at the top.
3. Wait for the player to press the F key.
4. Drop the bomb.
5. Make noise.

Program 14-4 Bombing under Keyboard Control

Do not type NEW. Leave Program 14-3 in memory.

```
100 REM PROGRAM 14-4
101 REM PLACE BOMB AT TOP
102 R=1
103 GOSUB 1000
104 REM PRESS F KEY TO DROP BOMB
105 CALL KEY(0,CH,FR)
106 IF CH<>70 THEN 135
RUN
```

Zero, not letter O

Remember to hold F down until the bomb starts falling. Program 14-4 always drops the bomb in column 16. This is fine if there is a target in that column, but it does not present much of a challenge. We would like to be able to move the bomb horizontally before dropping it. To do this, we need more control keys, namely:

- R to move the bomb right one column.
- L to move the bomb left one column.
- E to end the game.

A series of IF-THEN statements will determine if the player has pressed F, R, L, or E. For example, to make the computer branch to line 240 if the player presses F, we use

```
IF CH=70 THEN 240
```

70 is the code for F from Table 6-1.

To add 1 to C (column number) if the player presses the R key (code 82), we use the statements

```
180 IF CH<>82 THEN 200
190 C=C+1
200 ...
```

Making C larger by 1 moves the bomb right one column if we draw it with GOSUB 1000.

To subtract 1 from C if the player presses L (code 76), we use

```
200 IF CH<>76 THEN 220
210 C=C-1
220 ...
```


The idea is the same. The computer does line 210 only if CH=76.

We must also put a space on each side of the bomb to eliminate shadows when it moves.

Program 14-5 Moving a Bomb Before Dropping It

```
NEW
100 REM PROGRAM 14-5
110 CALL CLEAR
115 REM DRAW GROUND WITH TARGETS
120 GOSUB 2000
125 REM START BOMB IN COLUMN 16
130 C=16
140 R=1
150 GOSUB 1000
155 REM LOOK FOR COMMANDS FROM KEYBOARD
160 CALL KEY(0,CH,PR)
165 REM F KEY DROPS BOMB
170 IF CH=70 THEN 240 ← 70 is F.
175 REM R KEY MOVES BOMB RIGHT
180 IF CH<>82 THEN 200 ← 82 is R.
190 C=C+1
195 REM L KEY MOVES BOMB LEFT
200 IF CH<>76 THEN 220 ← 76 is L.
210 C=C-1
215 REM E KEY ENDS GAME
220 IF CH=69 THEN 9000 ← 69 is E.
230 GOTO 150
235 REM DROP BOMB
240 FOR R=1 TO 24
245 REM DRAW BOMB
250 GOSUB 1000
255 REM PRODUCE NOISE
260 CALL SOUND(300,-2,20)
265 REM ERASE BOMB
270 CALL HCHAR(R,C,32,5)
280 NEXT R
290 GOTO 9000
990 REM DRAW SMALL BOMB (*) WITH SPACES
1000 CALL HCHAR(R,C,32)
1010 CALL HCHAR(R,C+1,40)
1020 CALL HCHAR(R,C+2,42)
1030 CALL HCHAR(R,C+3,41)
1040 CALL HCHAR(R,C+4,32)
1050 RETURN
1990 REM DRAW GROUND WITH TARGETS
2000 CALL HCHAR(24,1,95,32)
2010 FOR C=5 TO 30 STEP 5
2020 CALL HCHAR(24,C,88)
2030 NEXT C
2040 RETURN
9000 END
RUN
```

When you press L or R, the bomb moves like an amoeba, expanding and contracting as it pushes itself along. As before, we can make the program repeat forever by changing line 290 to

```
290 GOTO 140
```

Now, when you type RUN, the program returns to line 140 after dropping a bomb. You must press the E key to exit.

If you keep pressing the L or R key, C will eventually go outside the 0 to 28 range, producing an error in CALL HCHAR. Remember that the bomb with a space on either side occupies five columns. To stop this from happening, we need statements that keep the player from moving the bomb too far. These statements are

```
182 REM KEEP BOMB ON SCREEN RIGHT
184 IF C=28 THEN 200

202 REM KEEP BOMB ON SCREEN LEFT
204 IF C=1 THEN 220
```

These lines restrict the bomb to starting in columns 1 through 28. Now the bomb seems to run into invisible barriers on both sides. You can keep pressing the direction key, but the bomb will not move any farther.

We can also change the scene so that the parentheses around the bomb represent a bomb bay (perhaps attached to an aircraft above the screen). Now only the bomb (the asterisk) falls, while the bay remains at the top. To do this, we must change the falling bomb to a single asterisk. The new lines are

```
250 CALL HCHAR(R,C+2,42) ←———— Places asterisk under the
                             bomb bay as it moves down.
270 CALL HCHAR(R,C+2,32) ←———— Erases the asterisk.
```

Note that the asterisk is in column C+2 if the bomb starts in column C.

Program 14-6 Bombs Away Game

```
NEW
100 REM PROGRAM 14-6
110 CALL CLEAR
115 REM DRAW GROUND WITH TARGETS
120 GOSUB 2000
125 REM PLACE BOMB IN 1ST ROW, 16TH COLUMN
130 C=16
140 R=1
150 GOSUB 1000
155 REM LOOK FOR KEYBOARD COMMANDS
160 CALL KEY(0,CH,PR)
165 REM F KEY DROPS BOMB
170 IF CH=70 THEN 260 ←———— 70 is F.
175 REM R KEY MOVES BOMB RIGHT
180 IF CH>82 THEN 210 ←———— 82 is R.
185 REM KEEP BOMB ON SCREEN RIGHT
190 IF C=28 THEN 210
200 C=C+1
205 REM L KEY MOVES BOMB LEFT
210 IF CH>76 THEN 240 ←———— 76 is L.
215 REM KEEP BOMB ON SCREEN LEFT
220 IF C=1 THEN 240
230 C=C-1
235 REM E KEY ENDS GAME
240 IF CH=69 THEN 9000 ←———— 69 is E.
250 GOTO 150
```

```

255 REM DROP BOMB
260 FOR R=1 TO 24
265 REM DRAW BOMB
270 CALL HCHAR(R,C+2,42)
275 REM PRODUCE NOISE
280 CALL SOUND(300,-2,20)
285 REM ERASE BOMB
290 CALL HCHAR(R,C+2,32)
300 NEXT R
310 GOTO 140
990 REM DRAW SMALL BOMB (*) WITH SPACES
1000 CALL HCHAR(R,C,32)
1010 CALL HCHAR(R,C+1,40)
1020 CALL HCHAR(R,C+2,42)
1030 CALL HCHAR(R,C+3,41)
1040 CALL HCHAR(R,C+4,32)
1050 RETURN
1990 REM DRAW GROUND WITH TARGETS
2000 CALL HCHAR(24,1,95,32)
2010 FOR C=5 TO 30 STEP 5
2020 CALL HCHAR(24,C,88)
2030 NEXT C
2040 RETURN
9000 END
RUN

```

We can move the bomb by pressing R (right) or L (left). We can also drop the bomb by pressing F and end the game by pressing E. We use

```
310 GOTO 140
```

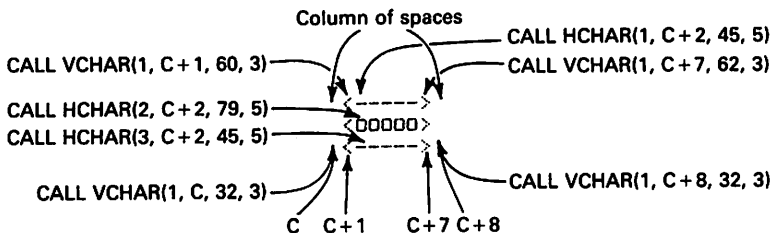
to make the game run continuously until the E key is pressed. Note that the new bomb appears in the same place from which the old bomb fell. The hole in the ground remains, although it is now only a single column wide. We can make the game more elaborate and more interesting by having an alien attack ship drop the bombs. The ship looks like this:

```

<----->
<00000>
<----->

```

The drawing is three lines tall and nine columns wide (there is a space on either side). To keep it on the screen, we must not start it beyond column 24. The HCHAR and VCHAR commands are



The bomb must start descending from row 4 to fall from under the ship.

Program 14-7 Alien Attack Ship Dropping Bombs

Do not type NEW. Leave Program 14-6 in memory.

```

100 REM PROGRAM 14-7
190 IF C=24 THEN 210
260 FOR R=4 TO 24

990 REM DRAW ATTACK SHIP WITH SPACE ON EACH SIDE
1000 CALL VCHAR(1,C,32,3) ← Spaces on left side
1010 CALL VCHAR(1,C+8,32,3) ← Spaces on right side
1020 CALL HCHAR(1,C+2,45,5) ← 3 center sections of ship
1030 CALL HCHAR(2,C+2,79,5) ← 3 center sections of ship
1040 CALL HCHAR(3,C+2,45,5) ← 3 center sections of ship
1050 CALL VCHAR(1,C+1,60,3) ← Left side of ship
1060 CALL VCHAR(1,C+7,62,3) ← Right side of ship
1070 RETURN
RUN

```

The spaceship appears to lose its sides briefly when you move it left or right. Remember, the only way to stop this program is by pressing E or FCTN and 4 (CLEAR). Also, remember that you must press F to drop the bomb, R to move the attack ship right, and L to move it left. When you press F, you will see a single bomb fall from the middle of the attack ship. You can make two bombs fall at a time by drawing two asterisks with

```

250 CALL HCHAR(J,C+4,42,2)
270 CALL HCHAR(J,C+4,32,2)

```

We can also produce an explosion with sound effects and flying debris when the bomb strikes the ground. The final scene looks like this:

```

<----->
<00000>
<----->

** ← 2 bombs falling

  \ \  / / ← Debris (after bombs land)
-----X-----X||||X-----X-----X-----

```

Program 14-8 Alien Attack Ship Dropping Bombs, with Explosion

Leave Program 14-7 in the computer. Do not type NEW.

```

100 REM PROGRAM 14-8
270 HCHAR(J,C+4,42,2)

290 HCHAR(J,C+4,32,2)
305 REM PRODUCE EXPLOSION
310 GOSUB 3000
320 B010 140

```

```
2995 REM EXPLOSION - FRAGMENTS, CLOUDS, AND NOISE
2997 REM DESTROY SOME GROUND
3000 CALL HCHAR(24,C+3,124,4)
3005 REM MAKE NOISE WHEN BOMB HITS
3010 CALL SOUND(1000,-6,10)
3015 REM SEND UP SOME DEBRIS
3020 CALL HCHAR(23,C+2,92,2)
3030 CALL HCHAR(23,C+5,47,2)
3035 REM PRODUCE EXPLOSION SOUND
3040 CALL SOUND(1000,-5,12)
3050 CALL SOUND(1000,-7,10)
3055 REM ERASE DEBRIS
3060 CALL HCHAR(23,C+2,32,6)
3070 RETURN
9000 END
RUN
```

15

DIAGONAL MOTION

So far, we have moved objects right, left, up, and down. We will now learn how to move things diagonally.

Let's draw a small eye creature <O> and move it right diagonally. All we have to do is change both the starting row and the starting column each time.

Program 15-1 Eye Creature Moving Right Diagonally

```
NEW
100 REM PROGRAM 15-1
110 CALL CLEAR
120 FOR J=1 TO 24
130 CALL HCHAR(J,J,60) ← Starting point changes from (1, 1) to
140 CALL HCHAR(J,J+1,79) ← (2, 2) to (3, 3), etc.
150 CALL HCHAR(J,J+2,62)
160 CALL SOUND(100,30000,30) ← Slows the computer down
170 CALL HCHAR(J,J,32,3) ← Erase the shadow with 3 spaces.
180 NEXT J
190 CALL CLEAR
RUN
```

We can even make a ball roll down a diagonal chute. We first draw the chute from two diagonals of slashes and then make a letter O move down the center.

Program 15-2 Ball Rolling Down Chute

```
NEW
100 REM PROGRAM 15-2
110 CALL CLEAR
115 REM DRAW SIDES
120 FOR J=1 TO 24
130 CALL HCHAR(J,J,92) ←
140 CALL HCHAR(J,J+2,92) ← 92 is \.
```

```

150 NEXT J
160 FOR N=1 TO 10 ← 10 balls roll down.
170 FOR J=1 TO 24
175 REM DRAW BALL
180 CALL HCHAR(J,J+1,79) ← 79 is O (letter).
190 CALL SOUND(100,30000,30)
195 REM ERASE BALL
200 CALL HCHAR(J,J+1,32)
210 CALL SOUND(100,30000,30) ← Slows the computer down

220 NEXT J
230 NEXT N
RUN

```

The ball appears to roll down the chute (or perhaps it is floating rapidly down a canal or traveling along a road). This program can produce many surprising effects if you change the character codes on lines 180 and 200. For example, try the following:

```

1) 180 CALL HCHAR(J,J+1,45)
   200 CALL HCHAR(J,J+1,95)

2) 180 CALL HCHAR(J,J+1,79)
   200 CALL HCHAR(J,J+1,48)

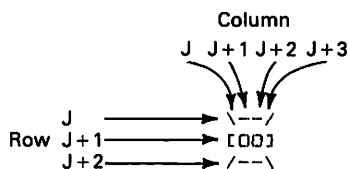
3) 180 CALL HCHAR(J,J+1,91)
   200 CALL HCHAR(J,J+1,93)

```

Feel free to experiment with other combinations.

Using the ideas from previous chapters, we can draw any object and move it diagonally. We must be careful, however, not to draw off the screen.

To make an alien creature move diagonally, we use the picture



By modifying the drawing commands in Program 13-4, we obtain the following statements to move the alien diagonally:

```

DRAW TOP    \--/

CALL HCHAR(J,J,92)
CALL HCHAR(J,J+1,45,2)
CALL HCHAR(J,J+3,47)

DRAW MIDDLE [00]

CALL HCHAR(J+1,J,91)
CALL HCHAR(J+1,J+1,79,2)
CALL HCHAR(J+1,J+3,93)

```

```

DRAW BOTTOM  /--\

CALL HCHAR(J+2,J,47)
CALL HCHAR(J+2,J+1,45,2)
CALL HCHAR(J+2,J+3,92)

```

We already know that planning and sketching on paper simplifies program writing. Note also that using separate subroutines for each idea or picture makes it easier to enter, correct, and modify a program.

Program 15-3 Alien Moving Diagonally, with Sound

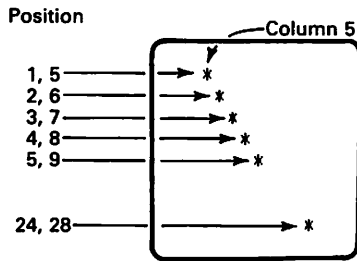
```

NEW
100 REM PROGRAM 15-3
110 CALL CLEAR
120 FOR J=1 TO 22  ← Limit is 22 since the alien is 3 lines tall.
125 REM DRAW ALIEN
130 GOSUB 1000
135 REM PRODUCE SOUND
140 CALL SOUND(20,-3,15)
150 CALL SOUND(5,125,15)
155 REM ERASE ALIEN
160 GOSUB 2000
170 NEXT J
180 GOTO 9000

990 REM DRAW ALIEN
999 REM DRAW TOP  \--/
1000 CALL HCHAR(J,J,92)
1010 CALL HCHAR(J,J+1,45,2)
1020 CALL HCHAR(J,J+3,47)
1025 REM DRAW MIDDLE [00J
1030 CALL HCHAR(J+1,J,91)
1040 CALL HCHAR(J+1,J+1,79,2)
1050 CALL HCHAR(J+1,J+3,93)
1055 REM DRAW BOTTOM  /--\
1060 CALL HCHAR(J+2,J,47)
1070 CALL HCHAR(J+2,J+1,45,2)
1080 CALL HCHAR(J+2,J+3,92)
1090 RETURN
1990 REM ERASE ALIEN
2000 FOR ROW=J TO J+2
2010 CALL HCHAR(ROW,J,32,4)
2020 NEXT ROW
2030 RETURN
9000 CALL CLEAR
RUN

```

We used FOR J=1 TO 22 to keep the three-line-high alien on the screen. To move an object from a column other than column #1, we must sketch the scene before writing the program. Let's move an asterisk diagonally beginning in column 5.



The FOR loop becomes

```
FOR J=1 TO 24
```

To start the asterisk in column 5, we must add 4 to each column value in CALL HCHAR.

Program 15-4 Asterisk Moving Diagonally from Column 5

```
NEW
100 REM PROGRAM 15-4
110 CALL CLEAR
120 FOR J=1 TO 24
130 CALL HCHAR(J,J+4,42)
140 CALL SOUND(200,30000,30)
150 CALL HCHAR(J,J+4,32)
160 NEXT J
170 CALL CLEAR
RUN
```

← Slows the computer down

← Erases picture

For the alien, which occupies more than one line, we can combine the ideas of Program 15-3 and 15-4. Here we need

```
FOR J=1 TO 22
```

to move the figure down.

We can also draw two different pictures of the alien to produce a pulsating figure. The second picture is

```
\==/
[??]
/==\
```

which requires only the following three CALL HCHARs:

```
== CALL HCHAR(J,J+1,61,2)
?? CALL HCHAR(J+1,J+1,64,2)
== CALL HCHAR(J+2,J+1,61,2)
```

Since we start the alien in column 5, we must add 4 to each column value in its picture.

**Program 15-5 Pulsating Alien Moving Right Diagonally
from Column 5**

```
NEW
100 REM PROGRAM 15-5
110 CALL CLEAR
120 FOR J=1 TO 22 ← Limit is 22 since the alien is 3 lines tall.
125 REM DRAW ALIEN MOVING DOWN
130 GOSUB 1000
135 REM PRODUCE SOUND
140 CALL SOUND(20,-3,15)
150 CALL SOUND(5,125,15)
155 REM DRAW 2ND PICTURE OF ALIEN
160 GOSUB 3000
165 REM PRODUCE SECOND SOUND
170 CALL SOUND(30,1000,22)
175 REM ERASE PICTURE
180 GOSUB 2000
190 NEXT J
200 GOTO 9000
990 REM DRAW ALIEN
999 REM DRAW TOP      \--/
1000 CALL HCHAR(J,J+4,92)
1010 CALL HCHAR(J,J+5,45,2)
1020 CALL HCHAR(J,J+7,47)
1025 REM DRAW MIDDLE [00]
1030 CALL HCHAR(J+1,J+4,91)
1040 CALL HCHAR(J+1,J+5,79,2)
1050 CALL HCHAR(J+1,J+7,93)
1055 REM DRAW BOTTOM  /--\
1060 CALL HCHAR(J+2,J+4,47)
1070 CALL HCHAR(J+2,J+5,45,2)
1080 CALL HCHAR(J+2,J+7,92)
1090 RETURN
1990 REM ERASE ALIEN
2000 FOR ROW=J TO J+2
2010 CALL HCHAR(ROW,J+4,32,4)
2020 NEXT ROW
2030 RETURN
2990 REM DRAW SECOND ALIEN
3000 CALL HCHAR(J,J+5,61,2)
3010 CALL HCHAR(J+1,J+5,64,2)
3020 CALL HCHAR(J+2,J+5,61,2)
3030 RETURN
9000 CALL CLEAR
RUN
```

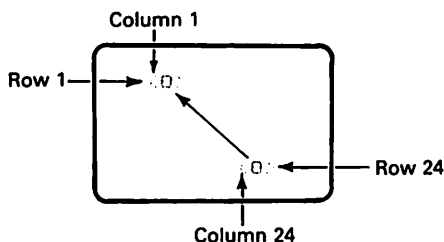
So far, objects have moved diagonally to the right. We can make objects move diagonally to the left by changing the FOR loop to

```
FOR J=24 TO 1 STEP -1
```

Program 15-6 Eye Creature Moving Left Diagonally

```
NEW
100 REM PROGRAM 15-6
110 CALL CLEAR
120 FOR J=24 TO 1 STEP -1
130 CALL HCHAR(J,J,60)
140 CALL HCHAR(J,J+1,79)
150 CALL HCHAR(J,J+2,62)
160 CALL SOUND(100,30000,30) ← Slows the computer down
170 CALL HCHAR(J,J,32,3) ← Erase the shadow with 3 spaces.
180 NEXT J
190 CALL CLEAR
RUN
```

The eye creatures follow a diagonal route with the starting point going from position 24,24 to 23,23 to 22,22 and so on, all the way to 1,1.



We can also redo Program 15-2 to make a tiny car (` `) move along a deserted highway. Think of this as a straight road seen from an airplane or helicopter.

Program 15-7 Car Traveling Along a Highway

```
NEW
100 REM PROGRAM 15-7
110 CALL CLEAR
115 REM DRAW SIDES OF HIGHWAY
120 FOR J=1 TO 24
130 CALL HCHAR(J,J,92) ← 92 is \.
140 CALL HCHAR(J,J+2,92) ← 92 is \.
150 NEXT J

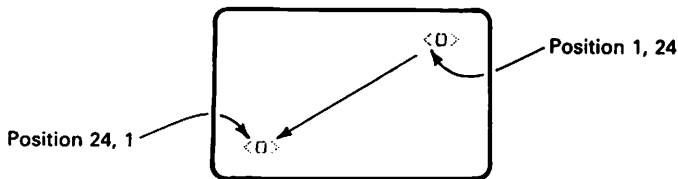
160 FOR N=1 TO 10 ← 10 cars
170 FOR J=24 TO 1 STEP -1
175 REM DRAW CAR
180 CALL HCHAR(J,J+1,96) ← 96 is ,.
190 CALL SOUND(100,30000,30)
195 REM ERASE CAR
200 CALL HCHAR(J,J+1,32)
210 CALL SOUND(100,30000,30) ← Slows the computer down
220 NEXT J
230 NEXT N
RUN
```

To reduce the car's speed, replace line 190 with

```
190 FOR K=1 TO 100  
195 NEXT K
```

You can also try the other character combinations we mentioned earlier on lines 180 and 200 to produce different effects.

Now, suppose we want to move the eye creature down diagonally, right to left. The drawing looks like this:



Let's go back to Program 2-8. We used

```
FOR J=1 TO 20  
PRINT TAB(J); "TONY"  
PRINT TAB(21-J); "MARY"  
NEXT J
```

We can use the same idea to draw the creatures, but with CALL HCHAR instead of TAB. J, 21-J, and 24-J change as follows.

J	21-J	24-J
1	20	23
2	19	22
3	18	21
19	2	5
20	1	4
21	—	3
22	—	2
23	—	1

Program 15-8 Eye Creature Moving Diagonally, Right to Left, from Top

```
NEW
100 REM PROGRAM 15-8
110 CALL CLEAR
120 FOR J=1 TO 24
130 CALL HCHAR(J,25-J,60)
140 CALL HCHAR(J,26-J,79)
150 CALL HCHAR(J,27-J,62)
160 CALL SOUND(100,30000,30)
170 CALL HCHAR(J,25-J,32,3)
180 NEXT J
190 CALL CLEAR
RUN
```

← Slows the computer down
← Erases the shadow by overprinting with 3 spaces

Again we must be careful to keep within the bounds of CALL HCHAR.

We can also make traffic move in both directions along a highway that extends from the bottom left-hand corner to the top right-hand corner. We draw the sides farther apart here to allow for two lanes.

Program 15-9 Traffic Moving Along a Highway

```
NEW
100 REM PROGRAM 15-9
110 CALL CLEAR
115 REM DRAW SIDES OF HIGHWAY
120 FOR J=1 TO 24
130 CALL HCHAR(25-J,J,47)
140 CALL HCHAR(25-J,J+3,47)
150 NEXT J
160 FOR N=1 TO 10
170 FOR J=1 TO 24
175 REM DRAW CARS IN DIFFERENT LANES
180 CALL HCHAR(25-J,J+2,39)
190 CALL HCHAR(J,26-J,39)
200 CALL SOUND(100,30000,30)
205 REM ERASE CARS
210 CALL HCHAR(25-J,J+2,32)
220 CALL HCHAR(J,26-J,32)
230 CALL SOUND(100,30000,30)
240 NEXT J
250 NEXT N
RUN
```

← 47 is '/'
← 39 is ' '

To avoid the illusion of the cars' turning around at the top and bottom, introduce a delay between repetitions; that is, add

```
242 FOR K=1 TO 500
244 NEXT K
```

To make the cars travel on the left-hand side of the road (British-style) instead of the right-hand side (American-style), insert

```
180 CALL HCHAR(25-J, J+1, 39)
190 CALL HCHAR(J, 27-J, 39)
210 CALL HCHAR(25-J, J+1, 32)
220 CALL HCHAR(J, 27-J, 32)
```

16

USING CUSTOM CHARACTERS

So far, we have drawn pictures using the characters on the keyboard. However, the TI allows you to do much more than this. You can actually create your own characters; this is like designing your own alphabet or defining your own set of shapes for children's building blocks. To create a character, we divide a square into eight rows of eight columns as shown in Figure 16-1. We thus end up with 64 tiny squares, each of which the computer can make either dark or light. This results in a tremendous number of different possible shapes, ranging from all light (a blank square) to all dark (a solid square).

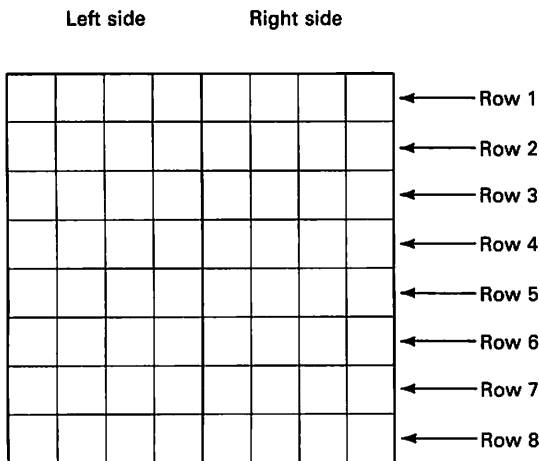
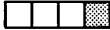
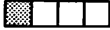
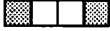


Figure 16-1 Grid for Graphics Characters

Of course, telling the computer to make 64 tiny squares either light or dark would require a tremendous amount of typing. Hence, we need a shorthand code like the Morse code used in telegraphy. The simplest approach is to use 0 for light and 1 for dark and to describe a square starting in the top left-hand corner and moving right and down, the way one reads a book. Thus we would start with either a 0 or a 1 for the top left-hand square, continue along row 1, then start row 2, and so on. A character would therefore consist of 64 digits, each of which is either 0 or 1.

Typing 64 digits is still a lot of work, so we compact the code by grouping sets of 4 digits as shown in Table 16-1. Note that we now start with the left block of row 1, move on to the right block of row 1, then the left block of row 2, and so forth. Because Table 16-1 contains 16 entries, it is called a *hexadecimal* code. This code is natural on all planets where inhabitants have 16 fingers instead of 10.

TABLE 16-1 COMPACT CODE FOR DESCRIBING CHARACTERS

Block	0-1 Code (0=Dark, 1=Light)	Compact Code
	0000	0
	0001	1
	0010	2
	0011	3
	0100	4
	0101	5
	0110	6
	0111	7
	1000	8
	1001	9
	1010	A
	1011	B
	1100	C
	1101	D
	1110	E
	1111	F

Look at the example of a stick figure character in Figure 16-2.

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Figure 16-2 Stick Figure

Using Table 16-1, we can describe this character in code as 18183C7EBD3C2424. Even in compact form, it is a lot of typing; it's longer than most credit card numbers or bank account numbers. We advise you to proceed cautiously when defining characters; be sure you have everything right. The process will remind you of constructing a large project from a child's building set; everything looks simple on the box, but it is far more complex in the reality of the living room floor. Once you have completed this translation, you must proceed as follows:

1. Put the compact code inside quotation marks and give it a string variable name. For example, we might call the stick figure ST\$; that is,

```
ST$="18183C7EBD3C2424"
```

2. Assign your new character a code number like the ones in Table 6-1. Use the numbers between 128 and 159 (inclusive), since the computer does not need these for its own purposes.

To assign a code number, use

```
CALL CHAR(NAME$,CODENUM)
```

in which NAME\$ is the name you gave the compact code and CODENUM is the code number you are assigning to this character. For example, we could define our stick figure with

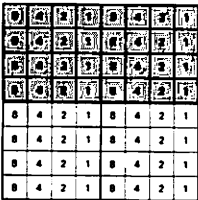
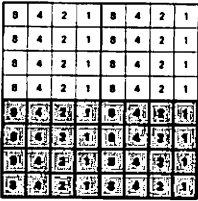
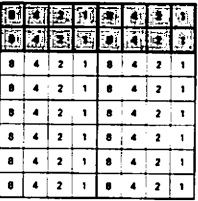
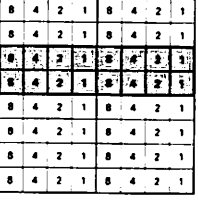
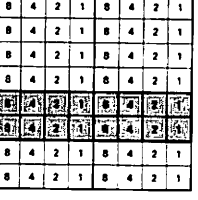
```
CALL CHAR(ST$,128)
```

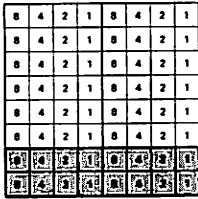
The stick figure is now character 128 as far as the computer is concerned. You can put it anywhere on the screen by using HCHAR or VCHAR with code 128.

Now you see why people rarely design their own alphabets and why creating building blocks is not as easy as you might think.

To get some practice with the character grid and coding methods, try some geometric shapes such as the ones listed in Table 16-2.

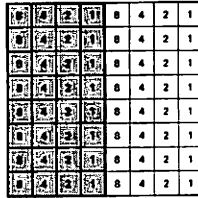
TABLE 16-2 COMMON GEOMETRIC CHARACTERS

Character	Description	Compact Code
	Top half dark	FFFFFFFF00000000
	Bottom half dark	00000000FFFFFFFF
	Top quarter dark	FFFF000000000000
	2nd quarter dark	0000FFFF00000000
	3rd quarter dark	00000000FFFF0000



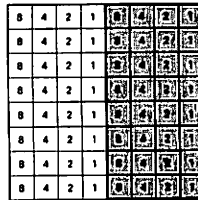
Bottom quarter dark

0000000000001111



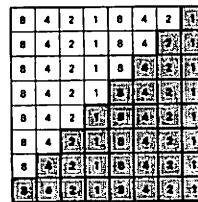
Left half dark

F0F0F0F0F0F0F0



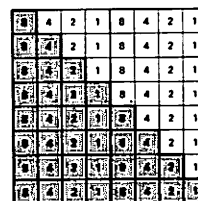
Right half dark

0F0F0F0F0F0F0F



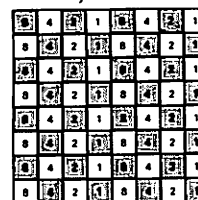
Bottom right-hand triangle

0103070F1F3F7FFF



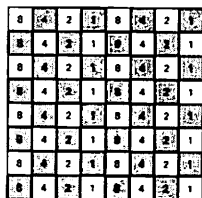
Bottom left-hand triangle

80C0E0F0F8FCFEFF



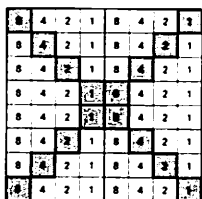
Left-hand checkerboard

AA55AA55AA55AA55



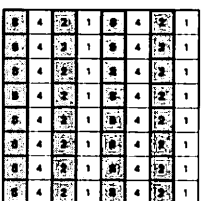
Right-hand checkerboard

55AA55AA55AA55AA



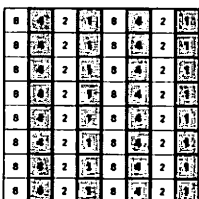
Large X

8142241818244281



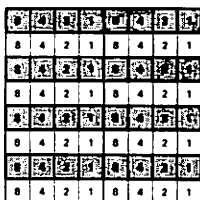
Left-hand vertical bars

AAAAAAAAAAAAAAAAAA



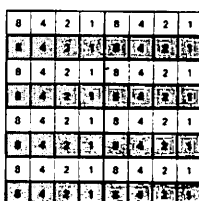
Right-hand vertical bars

5555555555555555



Horizontal bars on top

FF00FF00FF00FF00



Horizontal bars on bottom

00FF00FF00FF00FF

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Bottom triangle

FF8181818181FF

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Dark border

00000000183C7EFF

Let us now use the ideas in Program 7-3 to move the stick figure right.

Program 16-1 Stick Figure Moving Right

```

NEW
100 REM PROGRAM 16-1
105 REM CODE STICK FIGURE CHARACTER
110 ST$="18183C7EBD3C2424" ← Better check this before continuing.
115 REM DEFINE CHARACTER      It must be 16 characters long.
120 CALL CHAR(128,ST$)
130 CALL CLEAR
140 FOR C=1 TO 28
150 CALL HCHAR(5,C,128) ← Print the custom character (#128).
160 FOR K=1 TO 10
170 NEXT K
175 REM ERASE STICK FIGURE
180 CALL HCHAR(5,C,32)
190 NEXT C
200 CALL CLEAR
RUN

```

When defining characters, be sure each one consists of 16 compact codes (letters or digits). If you find it difficult to handle 16 codes at a time (it's like trying to remember and dial a 16-digit telephone number), try the following:

1. Divide the code into two 8-character sections and use & (*concatenation*) to combine them. For example, we could divide the stick figure into a top-half STT\$ and a bottom-half STB\$, then combine them as follows:

```

110 STT$="18183C7E"
112 STB$="BD3C2424"
120 CALL CHAR(128,STT$&STB$)

```

The & makes the computer put one string after another.

2. Check the length of the string with the built-in LEN (*length*) function. LEN(N\$) gives the number of characters in N\$, for all valid codes.

Neither approach is ideal, but both work better than counting on your fingers. You can reduce the figure's speed by changing line 160 to

```
160 FOR K=1 TO 50
```

A side profile would seem more reasonable than a head-on view. See if you can produce one on the 8×8 grid. To draw larger pictures, we must combine custom characters. For example, we could draw a rocket moving right as

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

Placing the picture in two consecutive grids, we get

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

We use Table 16-1 to convert the grids to the following codes:

Grid A	Grid B
3F	FB
3F	FC
BF	FE
FF	FF
FF	FF
BF	FE
3F	FC
3F	FS

The BASIC statements become

```
RBOT$="3F3FBFFFFFBF3F3F"  
RTOP$="FBFCFEFFFFFEFCFB"  
CALL CHAR(128,RBOT$)  
CALL CHAR(129,RTOP$)
```

We must keep the two characters together as they move; the first one is always in column C and the second in C+1.

Program 16-2 Rocket Moving Right

```
NEW  
100 REM PROGRAM 16-2  
105 REM CODE ROCKET CHARACTERS  
110 RBOT$="3F3FBFFFFFBF3F3F" ← The two codes should be the  
120 RTOP$="FBFCFEFFFFFEFCFB" ← same length.  
125 REM DEFINE CHARACTERS TO COMPUTER  
130 CALL CHAR(128,RBOT$)  
140 CALL CHAR(129,RTOP$)  
150 CALL CLEAR  
160 FOR C=1 TO 20  
170 CALL HCHAR(5,C,128)  
180 CALL HCHAR(5,C+1,129)  
190 FOR K=1 TO 10  
200 NEXT K  
210 CALL HCHAR(5,C,32,2) ← Prints two spaces over rocket  
220 NEXT C  
230 CALL CLEAR  
RUN
```

[illegible]

Grid A

[illegible]

Grid B

[illegible]

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Using Table 16-1, we get the codes

Grid A

0 6
0 6
6 F
F F
F F
F F
6 F
0 F

Grid B

6 0
6 0
F 6
F F
F F
F F
F 6
F 0

Grid C

0 F
6 F
F F
F F
F F
6 F
0 7
0 3

Grid D

F 0
F 6
F F
F F
F F
F 6
E 0
C 0

The BASIC statements are

```
A$="06066FFFFFFF6F0F"
B$="6060F6FFFFFFF6F0"
C$="0F6FFFFFFF6F0703"
D$="F0F6FFFFFFF6E0C0"
CALL CHAR(128,A$)
CALL CHAR(129,B$)
CALL CHAR(130,C$)
CALL CHAR(131,D$)
```

7 Fs in a row

We must hold the car together as it moves. We begin it in column 10 and put its rear end (A\$ and B\$) in row R and its front end (C\$ and D\$) in row R+1.

Program 16-3 Race Car Moving Down

```
NEW
100 REM PROGRAM 16-3
105 REM CODE CAR CHARACTERS
110 A$="06066FFFFFFF6F0F"
120 B$="6060F6FFFFFFF6F0"
130 C$="0F6FFFFFFF6F0703"
140 D$="F0F6FFFFFFF6E0C0"
145 REM DEFINE CHARACTERS
150 CALL CHAR(128,A$)
160 CALL CHAR(129,B$)
```

7 Fs

Remember, \$ is shifted 4;
FCTN 4 is clear!

```

170 CALL CHAR(130,C$)
180 CALL CHAR(131,D$)
190 CALL CLEAR
200 FOR R=1 TO 20
210 CALL HCHAR(R,10,128)
220 CALL HCHAR(R,11,129)
230 CALL HCHAR(R+1,10,130)
240 CALL HCHAR(R+1,11,131)
250 FOR K=1 TO 10
260 NEXT K
265 REM ERASE PICTURE
270 CALL HCHAR(R,10,32,2)
280 CALL HCHAR(R+1,10,32,2)
290 NEXT R
300 CALL CLEAR
RUN

```

To draw a pinchbug and move it right, we use

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

The grid drawing is

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

We convert this into codes as

Grid A	Grid B	Grid C	Grid D	Grid E	Grid F
E 3	B E	0 0	F F	F C	6 0
7 1	C 7	0 7	F F	F C	7 0
3 8	E 3	8 F	7 F	F F	F 9
1 C	7 1	D D	7 F	F F	F 9
7 F	F F	F 9	1 C	7 1	D D
7 F	F F	F 9	3 8	E 3	8 F
F F	F C	7 0	7 1	C 7	0 7
F F	F C	6 0	E 3	8 E	0 0

The BASIC commands are then

```

A$="E371381C7F7FFFFF"
B$="BEC7E371FFFFFCFC"
C$="00078FDDF9F97060"
D$="FFFF7F7F1C3871E3"
E$="FCFCFFFF71E3C78E"
F$="6070F9F9DD8F0700"
  
```

5 Fs

Program 16-4 Pinchbug Moving Right

```

NEW
100 REM PROGRAM 16-4
105 REM CODE PINCHBUG CHARACTERS
110 A$="E371381C7F7FFFFF"
120 B$="BEC7E371FFFFFCFC"
130 C$="00078FDDF9F97060"
140 D$="FFFF7F7F1C3871E3"
150 E$="FCFCFFFF71E3C78E"
160 F$="6070F9F9DD8F0700"
170 CALL CHAR(128,A$)
180 CALL CHAR(129,B$)
190 CALL CHAR(130,C$)
200 CALL CHAR(131,D$)
210 CALL CHAR(132,E$)
220 CALL CHAR(133,F$)
225 REM MOVE PINCHBUG RIGHT
230 CALL CLEAR
240 FOR C=1 TO 20
245 REM MOVE GRIDS A, B, C
250 CALL HCHAR(3,C,128)
260 CALL HCHAR(3,C+1,129)
270 CALL HCHAR(3,C+2,130)
275 REM MOVE GRIDS D, E, F
280 CALL HCHAR(4,C,131)
290 CALL HCHAR(4,C+1,132)
300 CALL HCHAR(4,C+2,133)
305 REM ERASE PINCHBUG
310 CALL SOUND(100,50000,10)
320 CALL HCHAR(3,C,32,3)
330 CALL HCHAR(4,C,32,3)
340 NEXT C
350 CALL CLEAR
360 END
  
```

5 Fs

Slows the computer down

The pinchbug moves right beginning in column 1 of row 3. Note that the program prints grids B and C starting in columns C+1 and C+2, respectively. Also, grids E and F start one row down.

To make the pinchbug open and shut its jaws, we draw two additional versions of grids C and F in the same place.

The pinchbug looks like

Picture #1

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

[illegible][illegible]

The grid layouts for the jaws are

Picture #2

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Picture #3

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

We can translate these pictures into codes

Picture #2		Picture #3	
Grid C	Grid F	Grid C	Grid F
00	64	0F	60
00	7C	0C	70
80	FC	8C	F8
DC	FC	DC	F8
FC	DC	F8	DC
FC	80	F8	8C
7C	00	70	0C
64	00	60	0F

So, we must add the following to Program 16-4:

```

100 REM PROGRAM 16-5
161 G$="000080DCFCFC7C64" ← Enter EDIT 160 to obtain a code
162 H$="647CFCFCDC800000" ← line for comparing lengths, then
163 I$="0F0C8C0CFBFB7060" ← enter NUM 161, 1 before inserting
164 J$="6070F8F8DC8C0C0F" ← these lines.

221 CALL CHAR(134,G$) ← Enter NUM 221, 1 before inserting
222 CALL CHAR(135,H$) ← these lines.
223 CALL CHAR(136,I$)
224 CALL CHAR(137,J$)

302 CALL HCHAR(3,C+2,134) ← Enter NUM 301, 1.
303 CALL HCHAR(4,C+2,135)

307 CALL HCHAR(3,C+2,136)
308 CALL HCHAR(4,C+2,137)

```

To slow the bug down, use

```

301 CALL SOUND(100,30000,10)
306 CALL SOUND(100,30000,10)

```

After making these changes, type RES to renumber the program.

Program 16-5 Pinchbug with Moving Jaws, Traveling Right

```

NEW
100 REM PROGRAM 16-5
105 REM CODE PINCHBUG CHARACTERS
110 A$="E371381C7F7FFFFF"
120 B$="8EC7E371FFFFFCFC"
130 C$="00078FDDF9F97060"
140 D$="FFFF7F7F1C3871E3"
150 E$="FCFCFFFF71E3C78E"
160 F$="6070F9F9DD8F0700"
170 G$="000080DCFCFC7C64"
180 H$="647CFCFCDC800000"
190 I$="0F0C8C0CFBFB7060"
200 J$="6070F8F8DC8C0C0F"
205 REM DEFINE CODES
210 CALL CHAR(A$,128)
220 CALL CHAR(B$,129)
230 CALL CHAR(C$,130)
240 CALL CHAR(D$,131)
250 CALL CHAR(E$,132)
260 CALL CHAR(F$,133)
270 CALL CHAR(G$,134)
280 CALL CHAR(H$,135)
290 CALL CHAR(I$,136)
300 CALL CHAR(J$,137)
305 REM MOVE PINCHBUG RIGHT
310 CALL CLEAR
320 FOR C=1 TO 20
325 REM MOVE GRIDS A, B, C

```



```

330 CALL HCHAR(3,C,128)
340 CALL HCHAR(3,C+1,129)
350 CALL HCHAR(3,C+2,130)
355 REM MOVE GRIDS D, E, F
360 CALL HCHAR(4,C,131)
370 CALL HCHAR(4,C+1,132)
380 CALL HCHAR(4,C+2,133)
390 CALL SOUND(100,30000,10)
400 CALL HCHAR(3,C+2,134) ← Draw picture #2.
410 CALL HCHAR(4,C+2,135)
420 CALL SOUND(100,30000,10)
430 CALL HCHAR(3,C+2,136) ← Draw picture #3.
440 CALL HCHAR(4,C+2,137)
445 REM ERASE PINCHBUG
450 CALL SOUND(100,30000,10) ← Slows the computer down
460 CALL HCHAR(3,C,32,3)
470 CALL HCHAR(4,C,32,3)
480 NEXT C
490 CALL CLEAR
RUN

```

To make the pinchbug's jaws move, we drew them in one stage of motion and then in another stage. We can do this with any object. To make motion more natural, we must draw more pictures with smaller changes between the stages. This is equivalent to increasing the frame rate on a motion picture camera, that is, taking more pictures per minute. Let us look at the stick figure of Program 16-1. We can simulate a dance by using the following drawings, all placed at the same location, one after another:

Picture #1

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Picture #2

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Picture #3

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Picture #4

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Picture #5

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

We can code the grids as

Grid A

1 8
1 8
3 C
F F
3 C
3 C
2 4
2 4

Grid B

1 8
1 9
3 E
7 C
8 C
3 C
4 4
8 8

Grid C

1 8
1 8
3 C
F F
3 C
3 C
2 4
1 2

Grid D

9 9
3 A
3 C
3 C
3 C
3 C
2 4
2 4

Grid E

9 8
5 8
3 C
3 E
3 D
3 C
4 2
8 1

To make the stick figure appear to dance, we draw pictures 1, 2, 3, 4, 5, 5, 4, 3, 2, and 1 all in the same place. Putting the character definition in a subroutine makes the movement easier to follow. First, let us make the stick figure dance on a floor in the middle of the screen.

Program 16-6 Stick Figure Dancing

```

NEW
100 REM PROGRAM 16-6
105 REM CODE CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 DRAW DANCE FLOOR
130 CALL HCHAR(11,1,45,32)
135 REM DRAW FIGURE DANCING
140 FOR C=1 TO 25
150 CALL HCHAR(10,15,128)
160 CALL HCHAR(10,15,129)
170 CALL HCHAR(10,15,130)
180 CALL HCHAR(10,15,131)
190 CALL HCHAR(10,15,132)
200 NEXT C
210 GOTO 9000
990 REM DEFINE CHARACTERS
1000 A$="18183CFF3C3C2424"
1010 B$="18193E7CBC3C4488"
1020 C$="18183CFF3C3C2412"
1030 D$="993A3C3C3C3C2424"
1040 E$="98583C3E3D3C4281"
1050 CALL CHAR(128,A$)
1060 CALL CHAR(129,B$)
1070 CALL CHAR(130,C$)
1080 CALL CHAR(131,D$)
1090 CALL CHAR(132,E$)
1100 RETURN
9000 CALL CLEAR
RUN

```

To slow the arm and leg motion and make the dance more natural (and provide sound — this is definitely modern dance or perhaps Jazzercise), use

```

155 CALL SOUND(20,290,10)
165 CALL SOUND(20,440,10)
175 CALL SOUND(20,360,10)
185 CALL SOUND(20,400,10)
195 CALL SOUND(20,330,10)

```

Because the computer needs time to produce sound, the pictures move slower. If the motion still does not look natural, change the timing in the SOUND statements from 20 to 30 or 40. However, changing these lengths may make the sounds too long or too strange for even the most modern audience. A second way to slow motion is to add do-nothing loops. By inserting

```

156 FOR K=1 TO 50
157 NEXT K
166 FOR K=1 TO 50
167 NEXT K
176 FOR K=1 TO 50

```

```

177 NEXT K
186 FOR K=1 TO 50
187 NEXT K
196 FOR K=1 TO 50
197 NEXT K

```

the dancing slows even further.

To experiment with these time delays, you could change each loop top

```

FOR K=1 TO 1
NEXT K

```

and insert

```

101 PRINT "WHAT DELAY DO YOU WANT"
102 INPUT "BETWEEN PICTURES? ":T
103 CALL "CLEAR"

```

? is FCN I.

To make the stick figure move right as it dances, we can change lines 150 through 200, placing C (the loop variable) in each of the CALL HCHAR statements.

Program 16-7 Stick Figure Dancing Across the Screen

Do not type NEW since we want to retain Program 16-6.

```

100 REM PROGRAM 16-7
101 CALL HCHAR(10,C,129)
102 CALL HCHAR(10,C,129)
103 CALL HCHAR(10,C,130)
104 CALL HCHAR(10,C,131)
105 CALL HCHAR(10,C,132)

155 CALL SOUND(20,290,10)
165 CALL SOUND(20,440,10)
175 CALL SOUND(20,360,10)
185 CALL SOUND(20,400,10)
195 CALL SOUND(20,330,10)
196 REM ERASE OLD FIGURE
197 CALL HCHAR(10,C,32,5)
RUN

```

We can make a larger figure perform on stage by using the drawings

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid G

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid H

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

These grid values are

Picture #2

Grid E	Grid F	Grid G	Grid H
C 3	C 3	0 7	E 0
C 2	4 3	0 7	E 0
B 2	4 1	0 7	E 0
9 F	F 9	0 7	E 0
9 0	0 9	0 C	3 0
F 7	E F	0 C	3 0
F 0	0 F	1 B	1 B
F F	F F	1 B	1 B

Picture #2

Grid E	Grid F	Grid G	Grid H
C 3	C 3	0 7	E 0
C 2	4 3	0 7	E 0
B 2	4 1	0 7	E 0
9 F	F 9	0 7	E 0
9 0	0 9	0 C	3 0
F 7	E F	0 C	3 0
F 0	0 F	1 B	1 B
F F	F F	1 B	1 B

Program 16-8 Torg and His Terrible Dance

```

NEW
100 REM PROGRAM 16-8
105 REM DEFINE CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
130 FOR C=1 TO 20
140 GOSUB 2000
150 CALL SOUND(20,340,20)
160 CALL SOUND(10,290,20)
170 FOR K=1 TO 40
180 NEXT K

```

```

190 GOSUB 3000
200 CALL SOUND(20,340,20)
210 CALL SOUND(10,290,20)
220 FOR K=1 TO 40
230 NEXT K
235 REM ERASE OLD PICTURE
240 CALL HCHAR(9,C,32,2)
250 CALL HCHAR(10,C,32,2)
260 NEXT C
270 GOTO 9000

995 REM DEFINE ALL THE CHARACTERS
998 REM PICTURE #1
1000 A$="0302021F15F0FAFF"
1010 B$="004040F8A80F5FFF"
1020 C$="C7C7C7A7A6060606"
1030 D$="E3E3E3E5606060"
1040 CALL CHAR(128,A$)
1050 CALL CHAR(129,B$)
1060 CALL CHAR(130,C$)
1070 CALL CHAR(131,D$)
1075 REM PICTURE #2
1080 E$="C3C2829F90F7F0FF"
1090 F$="C34341F909EF0FFF"
1100 G$="070707070C0C1B1B"
1110 H$="E0E0E0E030301B1B"
1120 CALL CHAR(132,E$)
1130 CALL CHAR(133,F$)
1140 CALL CHAR(134,G$)
1150 CALL CHAR(135,H$)
1160 RETURN
1995 REM DRAW PICTURE #1
2000 CALL HCHAR(9,C,128)
2010 CALL HCHAR(9,C+1,129)
2020 CALL HCHAR(10,C,130)
2030 CALL HCHAR(10,C+1,131)
2040 RETURN
2995 REM DRAW PICTURE #2
3000 CALL HCHAR(9,C,132)
3010 CALL HCHAR(9,C+1,133)
3020 CALL HCHAR(10,C,134)
3030 CALL HCHAR(10,C+1,135)
3040 RETURN
9000 CALL CLEAR
RUN

```

These codes must all be the same length.
 4 consecutive "07"s
 4 consecutive "E0"s

Grid A
 Grid B – 1 column right of A
 Grid C – 1 row under A
 Grid D – 1 column right of C
 Grid E
 Grid F – 1 column right of E
 Grid G – 1 row under E
 Grid H – 1 column right of G

Run Program 16-8. Design more steps for the figure, and add them by defining the characters in the 1000 routine and placing the dance instructions in routines 4000 – 4040, 5000 – 5040 and so on. You can animate many pictures by drawing variations of scenes at the same location. This is the way people produce cartoons, television logo displays, and animated arcade games.

17

COMBINING PROGRAMS

We are now ready to produce more complex scenes by combining earlier programs. For example, let us draw a master spaceship that can drop a bomb. The master ship looks like this:

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

To draw it, we need the following three grid layouts:

The codes are

Grid D

```
F F
E 7
E 7
F F
7 E
7 E
2 4
2 4
```

giving

```
D$="FFE7E7FF7E7E2424"
```

The program will let us move the master ship right or left, drop a bomb, and then move the bomb right or left as it descends. We will use the following control keys:

- E Stop the game (*exit*).
- L Move the master ship or bomb left.
- R Move the master ship or bomb right.
- D Drop a bomb.

After a bomb lands, we can return to controlling the master spaceship again.

Since this is a complex project, we will divide it into small programs (sometimes called *modules*), each of which does only one thing. Let us first write the program that moves the master ship. To avoid a shadow, we will put a space on each side. Since the ship with the spaces is five columns wide, it can start anywhere from column 1 through column 28. Moving the master ship involves the same procedure as moving the ship in Program 14-7. We will use CALL KEY(0,CH,PR) to check the keyboard along with values from Table 6-1 (E is 69; L, 76; R, 82; and D, 68).

Program 17-1 Moving the Master Spaceship

```
NEW
100 REM PROGRAM 17-1
105 REM CODE CUSTOM CHARACTERS
110 A$="3F408E8E403F0909"
120 B$="FF003C3C00FF2424"
130 C$="FC02717102FE9090"
135 REM DEFINE CHARACTERS
140 CALL CHAR(128,A$)
150 CALL CHAR(129,B$)
160 CALL CHAR(130,C$)
170 CALL CLEAR
```

Must be 16 characters
B\$ and C\$ must be the same length as A\$.

```

175 REM DRAW GROUND
180 CALL HCHAR(24,1,95,32)
185 REM START MASTER SHIP IN COLUMN 10
190 C=10
200 GOSUB 1000
205 REM LET PLAYER CONTROL MASTER SHIP
210 CALL KEY(0,CH,PR)
215 REM E KEY STOPS GAME (EXIT)
220 IF CH=69 THEN 9000 ← 69 is E.
225 REM R KEY MOVES SHIP RIGHT
230 IF CH>82 THEN 260 ← 82 is R.
235 REM KEEP SHIP ON SCREEN RIGHT
240 IF C=28 THEN 260
250 C=C+1
255 REM L KEY MOVES SHIP LEFT
260 IF C<>76 THEN 400 ← 76 is L.
265 REM KEEP SHIP ON SCREEN LEFT
270 IF C=1 THEN 400
280 C=C-1
295 REM D KEY DROPS BOMB
400 IF CH>68 THEN 200 ← 68 is D.
410 PRINT "DROP A BOMB"
420 GOTO 200
995 REM DRAW MASTER SPACESHIP
1000 CALL HCHAR(1,C,32)
1010 CALL HCHAR(1,C+1,128)
1020 CALL HCHAR(1,C+2,129)
1030 CALL HCHAR(1,C+3,130)
1040 CALL HCHAR(1,C+4,32)
1050 RETURN
9000 CALL CLEAR
RUN

```

Put 1 space on each side of ship.

When you press R or L, the master ship wriggles right or left like an insect or centipede. We left some room in the line numbering to allow for expansion. We could use RES later to resequence, but remember that it will number the subroutines continuously with the main program.

Test Program 17-1. Move the master ship left and right with L and R. Press E to exit. Enter RUN again. Press D. You should see the message DROP A BOMB. Be sure to hold E or D down for a while; the computer may not respond immediately. We will later write a section that actually drops a miniship; for now, all we have is a short substitute. We refer to this kind of temporary replacement for testing purposes as a *program stub*, and the approach of starting with the overall program and then adding more details is known as *top-down design*.

Now let's draw a bomb under the master ship. We must first determine where to place it.

A space

8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

Column C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Column C + 2

We start the bomb on line 2, just below the center of the master ship. To keep from confusing the master ship's column locations with those of the bomb, we use C for the start of the master ship and CB for the bomb (CB=C+2).

Program 17-2 Master Ship with Bomb Underneath

Do not type NEW since we want to retain Program 17-1.

```
100 REM PROGRAM 17-2
132 D$="FFE7E7FF7E7E2424"
165 CALL CHAR(131,D$)
```

Enter EDIT 130 before typing line 132.
This will put C\$ on the screen so you can be sure D\$ is the right length.

```
410 CB=C+2
420 CALL HCHAR(2,CB,131)
430 GOTO 130
RUN
```

Now when you type D, a bomb appears under the master ship. We are obviously well on our way to developing the complete program. The next section we must add is one

that makes the bomb fall. We will use the L and R keys to let the player guide the bomb as it descends. As usual, we must print a space over each picture to avoid a shadow. We will also add targets on the ground as in Program 14-7.

Program 17-3 Master Ship Attacking with Guided Bombs

Do not type NEW !

```

100 REM PROGRAM 17-3
415 REM MAKE BOMB FALL
420 FOR R=2 TO 24
425 REM DRAW AND ERASE BOMB
430 CALL HCHAR(R,CB,131)
435 REM PRODUCE SOUND
440 CALL SOUND(60,-2,10)
450 CALL HCHAR(R,CB,32)
460 CALL KEY(0,CH,PR)
465 REM R KEY MOVES BOMB RIGHT
470 IF CH<>82 THEN 500
475 REM KEEP BOMB ON SCREEN RIGHT
480 IF CB=30 THEN 500
490 CB=CB+1
495 REM L KEY MOVES BOMB LEFT
500 IF CH<>76 THEN 530
505 REM KEEP BOMB ON SCREEN LEFT
510 IF CB=3 THEN 530
520 CB=CB-1
530 NEXT R
540 GOTO 200

180 GOSUB 4000
3995 REM DRAW GROUND WITH TARGETS
4000 CALL HCHAR(24,1,95,32)
4010 FOR C=5 TO 30 STEP 5
4020 CALL HCHAR(24,C,88)
4030 NEXT C
4040 RETURN
RUN

```

Be sure to turn up the volume on your television.

Prints a space over the bomb

82 is R.

Restrict bomb to columns 3 through 30 to allow room for an explosion around it.

See how long it takes you to destroy all the targets. Since the bombs move slowly, you should be able to hit them with ease. You can make the bombs move faster by reducing the duration (the first item) in the CALL SOUND on line 440. You get only 23 chances to move a bomb, since we used

FOR R=2 TO 24

to control the descent.

Note the importance of dividing a large program into small programs (called *subroutines* or *modules*). You can then start with a rough outline of what you want and add more features as you progress. The best approach is always to use the same pattern. Assign each idea a group of line numbers, and add one new section at a time to your program. Test each feature to make sure it works.

You can add even more player control to the game. For example, we could let the player determine the bomb's speed. This involves changing the durations in the CALL

SOUND statements in lines 440, 450, and 460. Let us call the duration T and give it an initial value of 100. We will assign two new control keys as follows:

- F (*fast*) makes the bomb move faster.
- S (*slow*) makes the bomb move slower.

This works as follows:

1. Pressing F makes the computer divide T by 2, making the delay between pictures half as long (by halving the duration of the sounds).
2. Pressing S makes the computer multiply T by 2, making the delay between pictures twice as long (by doubling the duration of the sounds).

Program 17-4 contains the revisions required by introduce speed control.

Program 17-4 Spaceship Game with Speed Control

Do not type NEW!

```
100 REM PROGRAM 17-4
412 REM START DURATION AT 64
413 T=64

440 CALL SOUND(T,-2,10)

441 REM SPEED CONTROL (F DOUBLES, S HALVES)
442 IF CHK>70 THEN 446 ← 70 is F.
443 T=T/2 ←
446 IF CHK>83 THEN 450 ← 83 is S.
447 T=T*2 ←
RUN
```

Note that / means divide and * means multiply.

Run Program 17-4. Press D to drop a bomb, then press F several times. Be sure you hold it down for a while so the computer sees it. The bomb should speed up noticeably. Don't press F too many times or you will get a BAD VALUE in line 440 (the limits on duration are 1 to 4250).

Press D again to drop another bomb, and press S repeatedly. The bomb virtually stops moving, and the noise becomes longer and louder. In fact, it looks as though the bomb were suspended in air, trying to deafen the enemy (or game player) into submission. Furthermore, pressing F doesn't help; the bomb just hangs there and screams at you, like a phonograph record when the needle is stuck.

The problem is that the computer is spending so much time producing sounds that it hardly ever looks at the keyboard. Once you slow the bomb down, therefore, it is virtually impossible to get it moving again (and the noise is terrible). One way to keep this from happening is to keep the speed within reasonable limits. The following lines keep T between 4 and 256:

```

444 IF T>=4 THEN 446
445 T=4
448 IF T<=256 THEN 450
449 T=256

```

>= means "greater than or equal to."

<= means "less than or equal to."

We can produce an explosion when the bomb strikes the ground by inserting lines similar to those we used in Program 14-8. We must add

```

535 REM CREATE EXPLOSION WHEN BOMB HITS GROUND
540 GOSUB 5000
550 GOTO 200

4995 REM EXPLOSION - FRAGMENTS, SMOKE, AND NOISE
4997 REM DESTROY SOME GROUND
5000 CALL HCHAR(24,CB-2,124,4)
5005 REM MAKE NOISE WHEN BOMB HITS
5010 CALL SOUND(1000,-6,10)
5015 REM SEND UP SOME DEBRIS
5020 CALL HCHAR(23,CB-2,92,2)
5030 CALL HCHAR(23,CB,124)
5040 CALL HCHAR(23,CB+1,47,2)
5045 REM MAKE MORE EXPLOSION NOISE
5050 CALL SOUND(1000,-5,12)
5060 CALL SOUND(1000,-7,10)
5065 REM ERASE DEBRIS
5070 CALL HCHAR(23,CB-2,32,5)
5080 RETURN

```

The complete game then is as follows:

Program 17-5 Spaceship Game with Speed Control and Explosions

```

NEW
100 REM PROGRAM 17-1
105 REM CODE CUSTOM CHARACTERS
110 A$="3F408EBE403F0909"
120 B$="FF003C3C00FF2424"
130 C$="FC02717102FE9090"
132 D$="FFE7E7FF7E7E2424"
135 REM DEFINE CHARACTERS
140 CALL CHAR(128,A$)
150 CALL CHAR(129,B$)
160 CALL CHAR(130,C$)
165 CALL CHAR(131,D$)
170 CALL CLEAR
175 REM DRAW GROUND
180 GOSUB 4000

185 REM START MASTER SHIP IN COLUMN 10
190 C=10
200 GOSUB 1000
205 REM LET PLAYER CONTROL MASTER SHIP
210 CALL KEY(0,CH,PR)
215 REM E KEY STOPS GAME (EXIT)
220 IF CH=69 THEN 9000

```

Must be 16 characters
B\$ and C\$ must be the same length as A\$.

69 is E.

```

225 REM R KEY MOVES SHIP RIGHT
230 IF CHK>82 THEN 260 ← 82 is R.
235 REM KEEP SHIP ON SCREEN RIGHT
240 IF C=28 THEN 260
250 C=C+1
255 REM L KEY MOVES SHIP LEFT
260 IF C<76 THEN 400 ← 76 is L.
265 REM KEEP SHIP ON SCREEN LEFT
270 IF C=1 THEN 400
280 C=C-1
395 REM D KEY DROPS BOMB
400 IF CHK>68 THEN 200 ← 68 is D.
410 CB=C+2
412 REM START DURATION AT 64
413 T=64
415 REM MAKE BOMB FALL
420 FOR R=2 TO 24
425 REM DRAW AND ERASE BOMB
430 CALL HCHAR(R,CB,131)
435 REM PRODUCE SOUND
440 CALL SOUND(T,-2,10)
441 REM SPEED CONTROL (F DOUBLES, S HALVES)
442 IF CHK>70 THEN 446 ← 70 is F.
443 T=T/2
444 IF T<=4 THEN 446
445 T=4
446 IF CHK>83 THEN 450 ← 83 is S.
447 T=T*2
448 IF T<=256 THEN 450
449 T=256
450 CALL HCHAR(R,CB,32) ← Prints a space over the bomb
455 REM LOOK AT GUIDANCE COMMANDS
460 CALL KEY(O,CH,PR)
465 REM R KEY MOVES BOMB RIGHT
470 IF CHK>82 THEN 500 ← 82 is R.
475 REM KEEP BOMB ON SCREEN RIGHT
480 IF CB=30 THEN 500 ← Restrict bomb to columns 3 through 30
490 CB=CB+1 ← to allow room for an explosion around
495 REM L KEY MOVES BOMB LEFT ← it.
500 IF CHK>76 THEN 530
505 REM KEEP BOMB ON SCREEN LEFT
510 IF CB=3 THEN 530
520 CB=CB-1
530 NEXT R
535 REM CREATE EXPLOSION WHEN BOMB HITS GROUND
540 GOSUB 5000
550 GOTO 200

995 REM DRAW MASTER SPACESHIP
1000 CALL HCHAR(1,C,32)
1010 CALL HCHAR(1,C+1,128)
1020 CALL HCHAR(1,C+2,129)
1030 CALL HCHAR(1,C+3,130)
1040 CALL HCHAR(1,C+4,32) ← Put 1 space on each side of ship.
1050 RETURN
3995 REM DRAW GROUND WITH TARGETS
4000 CALL HCHAR(24,1,95,32)
4010 FOR C=5 TO 30 STEP 5
4020 CALL HCHAR(24,C,88)
4030 NEXT C
4040 RETURN
4995 REM EXPLOSION - FRAGMENTS, SMOKE, AND NOISE
4997 REM DESTROY SOME GROUND
5000 CALL HCHAR(24,CB-2,124,4)
5005 REM MAKE NOISE WHEN BOMB HITS
5010 CALL SOUND(1000,-6,10)

```



```
5015 REM SEND UP SOME DEBRIS
5020 CALL HCHAR(23,CB-2,92,2)
5030 CALL HCHAR(23,CB,124)
5040 CALL HCHAR(23,CB+1,47,2)
5045 REM MAKE MORE EXPLOSION NOISE
5050 CALL SOUND(1000,-5,12)
5060 CALL SOUND(1000,-7,10)
5065 REM ERASE DEBRIS
5070 CALL HCHAR(23,CB-2,32,5)
5080 RETURN
9000 CALL CLEAR
RUN
```

18

CONTROLLING MOTION IN ALL DIRECTIONS

Now let us create a game that involves moving an object right, left, up, or down. This requires four control keys instead of two; the obvious ones to use are the four arrow keys. Let us first draw a small bomb and start it in the center of the screen. We will guide it with the following keys:

E (up arrow)	Move bomb up 1 line.
S (left arrow)	Move bomb left 1 column.
D (right arrow)	Move bomb right 1 column.
X (down arrow)	Move bomb down 1 line.

Note that we must test in the program for E, S, D, and X, since we will use the arrow keys without FCTN. Also, we use Q (for *quit*) to end the game.
The bomb looks like this:

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The code for this is

```
2 4
4 2
B D
3 C
3 C
B D
4 2
2 4
```

Since we want to move the bomb in any direction, we surround it with spaces to eliminate shadows. Of course, this makes the picture three columns wide and three rows high.

Program 18-1 Moving a Bomb

```
NEW
100 REM PROGRAM 18-1
110 A$="2442BD3C3CBD4224"
120 CALL CHAR(128,A$)
130 CALL CLEAR
135 REM START BOMB IN CENTER
140 R=12
150 C=16
155 REM DRAW BOMB
160 GOSUB 1000
165 REM LET PLAYER MOVE BOMB
170 CALL KEY(0,CH,PR)
175 REM 0 (QUIT) KEY ENDS GAME
180 IF CH=81 THEN 9000
185 REM LEFT ARROW (S KEY) MOVES BOMB LEFT
190 IF CH>83 THEN 220
200 IF C=1 THEN 220
210 C=C-1
215 REM RIGHT ARROW (D KEY) MOVES BOMB RIGHT
220 IF CH<>68 THEN 250
230 IF C=30 THEN 250
240 C=C+1
245 REM UP ARROW (E KEY) MOVES BOMB UP
250 IF CH<>69 THEN 280
260 IF R=1 THEN 280
270 R=R-1
275 REM DOWN ARROW (X KEY) MOVES BOMB DOWN
280 IF CH<>88 THEN 160
285 REM KEEP BOMB PLUS SPACES ON SCREEN DOWN
290 IF R=22 THEN 160
300 R=R+1
310 GOTO 160
995 REM DRAW BOMB WITH SPACES
997 REM ROW OF SPACES
1000 CALL HCHAR(R+1,C,32,3)
1005 REM COLUMN OF SPACES
1010 CALL VCHAR(R,C+1,32,3)
1015 REM DRAW BOMB IN CENTER
1020 CALL HCHAR(R+1,C+1,128)
1030 RETURN
9000 CALL CLEAR
RUN
```

Be sure you press FCTN and P to type " before pressing Enter. If you press P and then ENTER, you will get an error message and will have to type the entire line over.

81 is Q.

83 is S (left-arrow).

68 is D (right-arrow). Bomb with spaces is 3 columns wide.

69 is E (up-arrow).

88 is X (down-arrow).

Each picture occupies four grids (two rows and two columns). The grid layouts and the codes are as follows:

Picture 1, Starship Moving Left

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Codes

Grids

A

B

C

D

0 0
0 0
0 0
0 0
1 0
3 0
3 0
F F

F F
F F
0 C
0 C
7 C
F F
F F

F F
3 3
3 0
1 0
0 0
0 0
0 0
0 0

F F
F F
7 C
0 C
0 C
0 C
F F
F F

Picture 2, Starship Moving Right

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid G

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid H

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Codes

Grids

E

F

G

H

F F
F F
3 0
3 0
3 0
3 E
F F
F F

0 0
0 0
0 0
0 0
0 8
0 C
C C
F F

F F
F F
3 E
3 0
3 0
3 0
F F
F F

F F
C C
0 C
0 8
0 0
0 0
0 0
0 0
0 0

Picture 3, Starship Moving Up

Grid I

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid J

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid K

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid L

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Codes

Grids	I	J	K	L
	0 1	8 0	C 3	C 3
	0 1	8 0	C 7	E 3
	0 7	E 0	C 7	E 3
	0 F	F 0	C 7	E 3
	0 1	8 0	F F	F F
	0 1	8 0	F F	F F
	0 3	C 0	C 3	C 3
	0 3	C 0	C 3	C 3

Picture 4, Starship Moving Down

Grid M

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid N

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid P

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid Q

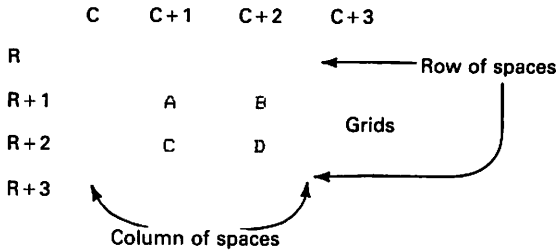
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Codes

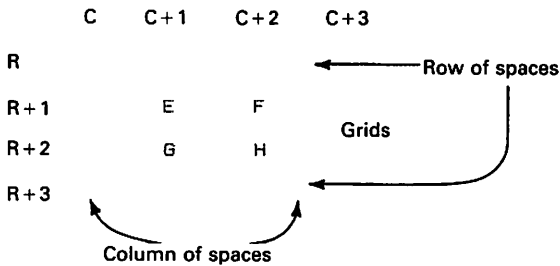
Grids	M	N	P	Q
	C 3	C 3	O 3	C 0
	C 3	C 3	O 3	C 0
	F F	F F	O 1	B 0
	F F	F F	O 1	B 0
	C 7	E 3	O F	F 0
	C 7	E 3	O 7	E 0
	C 7	E 3	O 1	B 0
	C 3	C 3	O 1	B 0

Now the program must not only move the spaceship in the proper direction and keep it on the screen but must also draw the appropriate picture. We must have spaces all around the ship at all times to avoid shadows. As in Program 18-1, we will use the arrow keys to move the ship. Since it is larger than the bomb, we must reduce the upper limits on the starting row and column. Be sure to keep the starship in the correct rows and columns so that the grid pictures and the spaces stay together. For each starship, the R and C values are as follows:

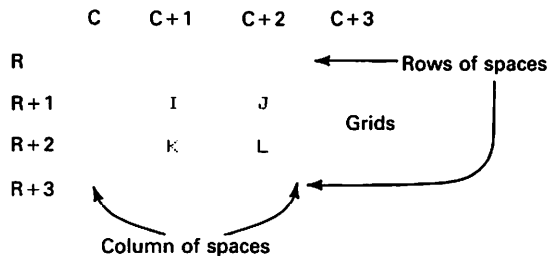
Starship Moving Left



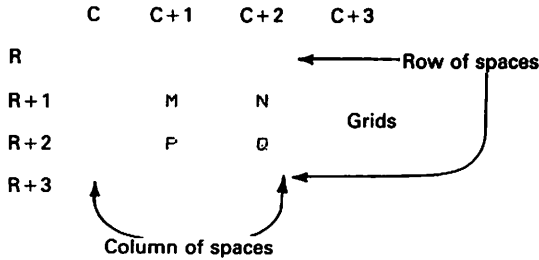
Starship Moving Right



Starship Moving Up

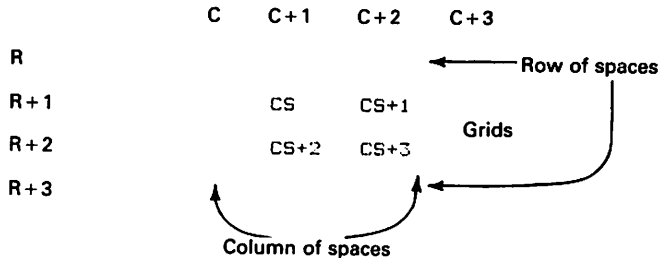


Starship Moving Down



We can generalize the starship picture as follows, since the geometry is always the same:

Generalized Starship



The only thing that differs in the four pictures is the starting character number CS; CS=128 for the starship moving right, 132 for left, 136 for up, and 140 for down.

Program 18-2 Starship Moving Anywhere

```

NEW
100 REM PROGRAM 18-2
105 REM DEFINE ALL STARSHIPS
110 GOSUB 2000
120 CALL CLEAR
125 REM START STARSHIP IN CENTER OF SCREEN
130 R=10
140 C=12
145 REM DRAW STARSHIP MOVING UP INITIALLY
150 CS=136
160 GOSUB 1000
165 REM WAIT FOR KEYBOARD COMMANDS
170 CALL KEY(O,CH,PR)
175 REM Q (QUIT) KEY ENDS GAME
180 IF CH=81 THEN 9000 ← 81 is Q.
185 REM LEFT ARROW (S KEY) MOVES STARSHIP LEFT
190 IF CH<>83 THEN 230 ← 83 is S (left-arrow).
200 IF C=1 THEN 220
210 C=C-1

```

```

215 REM STARTING CHARACTER FOR STARSHIP LEFT
220 CS=128
225 REM RIGHT ARROW (D KEY) MOVES STARSHIP RIGHT
230 IF CH<>68 THEN 270 ← 68 is D (right-arrow).
235 REM KEEP STARSHIP PLUS SPACES ON SCREEN RIGHT
240 IF C=29 THEN 260 ← Starship with spaces is 4
250 C=C+1 columns wide.
255 REM STARTING CHARACTER FOR STARSHIP RIGHT
260 CS=132
265 REM UP ARROW (E KEY) MOVES STARSHIP UP
270 IF CH<>69 THEN 310 ← 69 is E (up-arrow).
280 IF R=1 THEN 300
290 R=R-1
295 REM STARTING CHARACTER FOR STARSHIP UP
300 CS=136
305 REM DOWN ARROW (X KEY) MOVES STARSHIP DOWN
310 IF CH<>88 THEN 350 ← 88 is X (down-arrow).
320 IF R=21 THEN 340 ← Starship with spaces is 4
330 R=R+1 rows high.
335 REM STARTING CHARACTER FOR SPACESHIP DOWN
340 CS=140
345 REM DRAW STARSHIP
350 GOSUB 1000
360 GOTO 170

995 REM GENERALIZED STARSHIP PICTURE
998 REM PUT SPACES AROUND PICTURE
1000 CALL HCHAR(R,C+1,32,2)
1010 CALL HCHAR(R+3,C+1,32,2)
1020 CALL VCHAR(R+1,C,32,2)
1030 CALL VCHAR(R+1,C+3,32,2)
1035 REM PUT STARSHIP IN CENTER (CS IS STARTING CHARACTER)
1040 CALL HCHAR(R+1,C+1,CS)
1050 CALL HCHAR(R+1,C+2,CS+1)
1060 CALL HCHAR(R+2,C+1,CS+2)
1070 CALL HCHAR(R+2,C+2,CS+3)
1080 RETURN
1995 REM DEFINE STARSHIP MOVING LEFT
2000 A$="00000000103033FF" ← 8 zeros
2010 B$="FFFF0C0C0C7CFFFF"
2020 C$="FF33301000000000" ← 9 zeros
2030 D$="FFFF7C0C0C0CFFFF"
2040 CALL CHAR(128,A$)
2050 CALL CHAR(129,B$)
2060 CALL CHAR(130,C$)
2070 CALL CHAR(131,D$)
2075 REM DEFINE STARSHIP MOVING RIGHT
2080 E$="FFFF3030303EFFFF"
2090 F$="00000000080CCCF" ← 9 zeros
2100 G$="FFFF3E303030FFFF"
2110 H$="FFCC0C0800000000" ← 8 zeros
2120 CALL CHAR(132,E$)
2130 CALL CHAR(133,F$)
2140 CALL CHAR(134,G$)
2150 CALL CHAR(135,H$)
2155 REM DEFINE STARSHIP MOVING UP
2160 I$="0101070F01010303"
2170 J$="8080E0F08080C0C0"
2180 K$="C3C7C7C7FFFC3C3"
2190 L$="C3E3E3E3FFFC3C3"
2200 CALL CHAR(136,I$)
2210 CALL CHAR(137,J$)
2220 CALL CHAR(138,K$)
2230 CALL CHAR(139,L$)
2235 REM DEFINE STARSHIP MOVING DOWN
2240 M$="C3C3FFFFFC7C7C3"
2250 N$="C3C3FFFFE3E3E3C3"

```

```

2260 P$="030301010F070101"
2270 Q$="C0C0B0B0F0E0B0B0"
2280 CALL CHAR(140,M$)
2290 CALL CHAR(141,N$)
2300 CALL CHAR(142,P$)
2310 CALL CHAR(143,Q$)
2320 RETURN
9000 CALL CLEAR
RUN

```

Run Program 18-2 and move the starship around the screen. Move it all the way to the right, left, top, and bottom. It should not budge if you try to move it off the screen.

Each starship picture is four rows tall and four columns wide, so we must restrict R to 21 and C to 29.

Note how we got to this point. We started by making things move in different directions. Then we introduced control keys and moved things left and right. Next we moved a single picture of a bomb in four directions. Here we extended the program to handle an object that looks different, depending on which way it is heading. As we learned in Chapter 17, writing a complex program is much simpler if we divide it into small parts and make each part work.

If the program is modular, you can use sections of it later with minor changes. Although Program 18-2 is long, it follows naturally from Program 18-1. Once we had a program that moved a bomb in all four directions, we could easily extend it to moving four different pictures. Note how much of the main program (lines 100 through 995) stayed the same, since we put the pictures in subroutines.

To add sound, we could insert

```

1032 CALL SOUND(300,-2,20)

```

We can modify Program 18-2 to scatter objects randomly for the starship to hunt down. We will use the bomb from Program 18-1 as the target and the ideas from Program 10-3 to scatter it. The random scattering will be another subroutine, starting at line 3000.

Program 18-3 Starship Moving Anywhere, with Sound and Targets

Do not type NEW since we want to keep Program 18-2.

```

100 REM PROGRAM 18-3
121 REM PLACE RANDOM TARGETS ON SCREEN
122 GOSUB 3000

2995 REM PLACE RANDOM BOMBS ON SCREEN
2998 REM DEFINE BOMBS
3000 R$="2442BD3C3C8D4224"
3010 CALL CHAR(144,R$)
3020 FOR J=1 TO 10
3025 REM FIND RANDOM ROW AND COLUMN
3030 R=INT(RND*23)+1
3040 C=INT(RND*31)+1
3050 CALL HCHAR(R,C,144)
3060 NEXT J
3070 RETURN
RUN

```

Run Program 18-3 and see how long it takes you to destroy all the objects. Because of the spaces around the starship, it does not have to strike an object. The object disappears when the starship nears it. The starship appears to be surrounded by one of those invisible destructive shields made so popular by science fiction writers.

Note how convenient GOSUB and RETURN are. We can add new features easily from other programs by just assigning them some line numbers, adding a RETURN at the end, and putting a GOSUB in the main program. You can, in fact, add more features, such as more sounds or smoke trailing behind the starship. You are limited only by the amount of memory in your computer.

In Program 18-3, we must press a key to move the starship. This is unrealistic because a real vehicle has momentum. Once it is moving, it keeps going until you apply thrust in another direction. This is particularly true for a starship moving outside the atmosphere with neither gravity nor friction to slow it down.

Let us modify our game so that the starship maintains its momentum between commands. Instead of moving the starship, the command keys will simply establish its direction (DI\$): UP, DOWN, LEFT, or RIGHT. The starship will then move continuously in the DI\$ direction. The lines from 190 on are now

```

185 REM LEFT ARROW (S KEY) MAKES DIRECTION LEFT
190 IF CHK>83 THEN 230 ← 83 is S (left-arrow).
200 DI$="L"
210 CS=128
220 GOTO 340
225 REM RIGHT ARROW (D KEY) MAKES DIRECTION RIGHT
230 IF CHK>68 THEN 270 ← 68 is D (right-arrow).
240 DI$="R"
250 CS=132
260 GOTO 340
265 REM UP ARROW (E KEY) MAKES DIRECTION UP
270 IF CHK>69 THEN 310 ← 69 is E (up-arrow).
280 DI$="U"
290 CS=136
300 GOTO 340
305 REM DOWN ARROW (X KEY) MAKES DIRECTION DOWN
310 IF CHK>88 THEN 340 ← 88 is X (down-arrow).
320 DI$="D"
330 CS=140
335 REM MOVE STARSHIP IN DIRECTION DI$
338 REM MOVE STARSHIP LEFT
340 IF DI$<>"L" THEN 370
345 REM KEEP STARSHIP ON SCREEN LEFT
350 IF C=1 THEN 370
360 C=C-1
370 IF DI$<>"R" THEN 400
375 REM KEEP STARSHIP ON SCREEN RIGHT
380 IF C=29 THEN 400
390 C=C+1
395 REM MOVE STARSHIP UP
400 IF DI$<>"U" THEN 430
405 REM KEEP STARSHIP ON SCREEN UP
410 IF R=1 THEN 430
420 R=R-1
425 REM MOVE STARSHIP DOWN
430 IF DI$<>"D" THEN 460
435 REM KEEP STARSHIP ON SCREEN DOWN
440 IF R=21 THEN 460
450 R=R+1
460 GOSUB 1000
470 GOTO 170

```

We must also start the starship moving by inserting

```
155 DI$="U"
```

Program 18-4 Starship Game with Momentum

```
NEW
100 REM PROGRAM 18-4
105 REM DEFINE ALL STARSHIPS
110 GOSUB 2000
120 CALL CLEAR
125 REM START STARSHIP IN CENTER OF SCREEN
130 R=10
140 C=12
145 REM DRAW STARSHIP MOVING UP INITIALLY
150 CS=136
155 DI$="U"
160 GOSUB 1000
165 REM WAIT FOR KEYBOARD COMMANDS
170 CALL KEY(0,CH,PR)
175 REM Q (QUIT) KEY ENDS GAME
180 IF CH=81 THEN 9000 ← 81 is Q.
185 REM LEFT ARROW (S KEY) MAKES DIRECTION LEFT
190 IF CH<>83 THEN 230 ← 83 is S (left-arrow).
200 DI$="L"
210 CS=128
220 GOTO 340
225 REM RIGHT ARROW (D KEY) MAKES DIRECTION RIGHT
230 IF CH<>68 THEN 270 ← 68 is D (right-arrow).
240 DI$="R"
250 CS=132
260 GOTO 340
265 REM UP ARROW (E KEY) MAKES DIRECTION UP
270 IF CH<>69 THEN 310 ← 69 is E (up-arrow).
280 DI$="U"
290 CS=136
300 GOTO 340
305 REM DOWN ARROW (X KEY) MAKES DIRECTION DOWN
310 IF CH<>88 THEN 340 ← 88 is X (down-arrow).
320 DI$="D"
330 CS=140
335 REM MOVE STARSHIP IN DIRECTION DI$
338 REM MOVE STARSHIP LEFT
340 IF DI$<>"L" THEN 370
345 REM KEEP STARSHIP ON SCREEN LEFT
350 IF C=1 THEN 370
360 C=C-1
370 IF DI$<>"R" THEN 400
375 REM KEEP STARSHIP ON SCREEN RIGHT
380 IF C=29 THEN 400
390 C=C+1

395 REM MOVE STARSHIP UP
400 IF DI$<>"U" THEN 430
405 REM KEEP STARSHIP ON SCREEN UP
410 IF R=1 THEN 430
420 R=R-1
425 REM MOVE STARSHIP DOWN
430 IF DI$<>"D" THEN 460
435 REM KEEP STARSHIP ON SCREEN DOWN
440 IF R=21 THEN 460
450 R=R+1
460 GOSUB 1000
470 GOTO 170
```

```

995 REM GENERALIZED STARSHIP PICTURE
998 REM PUT SPACES AROUND PICTURE
1000 CALL HCHAR(R,C+1,32,2) ← 2 HCHARs
1010 CALL HCHAR(R+3,C+1,32,2) ← 2 HCHARs
1020 CALL VCHAR(R+1,C,32,2) ← 2 VCHARs
1030 CALL VCHAR(R+1,C+3,32,2) ← 2 VCHARs
1035 REM PUT STARSHIP IN CENTER (CS IS STARTING CHARACTER)
1040 CALL HCHAR(R+1,C+1,CS)
1050 CALL HCHAR(R+1,C+2,CS+1)
1060 CALL HCHAR(R+2,C+1,CS+2)
1070 CALL HCHAR(R+2,C+2,CS+3)
1080 RETURN
1995 REM DEFINE STARSHIP MOVING LEFT
2000 A$="00000000103033FF" ← 8 zeros
2010 B$="FFFF0C0C0C7CFFFF"
2020 C$="FF33301000000000" ← 9 zeros
2030 D$="FFFF7C0C0C0CFFFF"
2040 CALL CHAR(128,A$)
2050 CALL CHAR(129,B$)
2060 CALL CHAR(130,C$)
2070 CALL CHAR(131,D$)
2075 REM DEFINE STARSHIP MOVING RIGHT
2080 E$="FFFF3030303EFFFF"
2090 F$="000000000B0CCCF" ← 9 zeros
2100 G$="FFFF3E303030FFFF"
2110 H$="FFCC0C0B00000000" ← 8 zeros
2120 CALL CHAR(132,E$)
2130 CALL CHAR(133,F$)
2140 CALL CHAR(134,G$)
2150 CALL CHAR(135,H$)
2155 REM DEFINE STARSHIP MOVING UP
2160 I$="0101070F01010303"
2170 J$="80B0E0F0B0B0C0C0"
2180 K$="C3C7C7C7FFFC3C3"
2190 L$="C3E3E3E3FFFC3C3"
2200 CALL CHAR(136,I$)
2210 CALL CHAR(137,J$)
2220 CALL CHAR(138,K$)
2230 CALL CHAR(139,L$)

2235 REM DEFINE STARSHIP MOVING DOWN
2240 M$="C3C3FFFFFC7C7C3"
2250 N$="C3C3FFFFE3E3E3C3"
2260 P$="030301010F070101"
2270 Q$="C0C0B0B0F0E0B0B0"
2280 CALL CHAR(140,M$)
2290 CALL CHAR(141,N$)
2300 CALL CHAR(142,P$)
2310 CALL CHAR(143,Q$)
2320 RETURN
2995 REM SCATTER TARGETS RANDOMLY
2998 REM DEFINE THE BOMBS
3000 R$="2442BD3C3CBD4224"
3010 CALL CHAR(144,R$)
3020 FOR J=1 TO 10
3025 REM FIND RANDOM ROW, COLUMN
3030 R=INT(RND*23)+1
3040 C=INT(RND*31)+1
3050 CALL HCHAR(R,C,144)
3060 NEXT J
3070 RETURN
9000 CALL CLEAR
RUN

```

19

TARGET PRACTICE

Let us draw an artillery gun firing a shell. The shell first rises at an angle for a while and then falls toward the target. The artillery gun looks like this:

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

We convert these grids into the following codes:

A	B	C	D
0 0	0 0	F C	F F
0 0	0 0	F C	F F
0 0	0 F	F 0	F F
0 0	3 F	C 0	F F
0 0	F F	0 0	1 8
0 0	F 0	0 0	3 C
0 F	F 0	0 0	3 C
F F	C 0	0 0	1 8

The BASIC statements are

```

A$="000000000000307FF"
B$="00030F3FFFCFC0C0"
C$="FFFCFC0000000000"
D$="FFFFF183CC18"
  
```

10 zeros
 9 zeros
 8 Fs

The target (a factory with smokestacks) looks like this:

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid G

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid H

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid I

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid J

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid K								Grid L								Grid M							
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

The grids are

E	F	G	H	I	J	K	L	M
0 1	8 1	8 0	7 F	F F	F F	C A	8 2	A 3
0 1	8 1	8 0	F F	F F	F 3	C 0	0 0	0 3
0 1	8 1	8 0	C 0	0 0	0 3	C 0	0 0	0 3
0 1	8 1	8 0	C 0	0 0	0 3	C 0	3 8	0 3
0 C	F F	C 0	C A	8 2	A 3	C 0	3 8	0 3
0 F	F F	F 0	C 5	4 1	5 3	C 0	3 8	0 3
1 F	F F	F 8	C A	8 2	A 3	F F	F F	F F
3 F	F F	F C	C D	4 1	5 3	F F	F F	F F

producing

```

E$="01010101030F1F3F"
F$="81818181FFFFFFFF" ← 8 Fs
G$="80808080C0F0F8FC"
H$="7FFFC0C0C0C5CADD"
I$="FFFF000082418241"
J$="FFFC0703A353A353"
K$="CACC0C0C0C0C0FFF"
L$="8200003B3B3BFFFF"
M$="A0303030303FFFF"

```

The shell looks like this:

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

(producing N\$="183C7EFFFFFFFFF") 10 Fs

when it is rising and like this:

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

(producing P\$="FFFFFFFFF7E3C18") 10 Fs

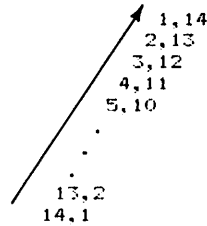
when it is falling.

You may want to define the substrings

```
FB$="FFFFFFF" ← 8 Fs
ZB$="0000000" ← 8 zeros
```

and use concatenation (&) to reduce the total amount of typing.

Let us start by making the shell rise diagonally from the lower left-hand corner to the upper right-hand corner. Its path is



8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The shell must travel from row 14 to row 1 and, at the same time, from column 1 to column 15. The lines that produce this motion are

```
FOR R=14 TO 1 STEP -1
  C=15-R
  CALL HCHAR(R,C,141)
NEXT R
```

Program 19-1 Shell Rising

```

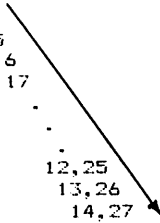
NEW
100 REM PROGRAM 19-1
105 REM DEFINE RISING SHELL
110 N$="183C7EFFFFFFFF" ← 10 Fs
120 CALL CHAR(141,N$)
130 CALL CLEAR
135 REM MOVE SHELL UP DIAGONALLY
140 FOR R=14 TO 1 STEP -1
150 C=15-R
160 CALL HCHAR(R,C,141)
165 REM PRODUCE SHELL NOISE
170 CALL SOUND(20,450,10)
175 REM ERASE SHELL
180 CALL HCHAR(R,C,32)
190 NEXT R
200 CALL CLEAR
RUN

```

The path of the falling shell is

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

1, 14
 2, 15
 3, 16
 4, 17
 .
 .
 .
 12, 25
 13, 26
 14, 27



Here the column number is always the row number plus 13.

Program 19-2 Shell Falling

```
NEW
100 REM PROGRAM 19-2
105 REM DEFINE FALLING SHELL
110 P$="FFFFFFFFF7E3C18" ← 10 Fs
120 CALL CHAR(142,P$)
130 CALL CLEAR
135 REM MOVE SHELL UP DIAGONALLY
140 FOR R=1 TO 14
150 C=R+13
160 CALL HCHAR(R,C,142)
165 REM PRODUCE SHELL NOISE
170 CALL SOUND(20,450,10)
175 REM ERASE SHELL
180 CALL HCHAR(R,C,32)
190 NEXT R
200 CALL CLEAR
RUN
```

Combining Programs 19-1 and 19-2 makes the shell rise and then fall.

Program 19-3 Shell Rising and Falling

```
NEW
100 REM PROGRAM 19-3
105 REM DEFINE BOTH SHELLS
110 GOSUB 1000
120 CALL CLEAR
125 REM MOVE SHELL UP DIAGONALLY
130 FOR R=14 TO 1 STEP -1
140 C=15-R
150 CALL HCHAR(R,C,141)
155 REM PRODUCE SHELL NOISE
160 CALL SOUND(20,450,10)
165 REM ERASE SHELL
170 CALL HCHAR(R,C,32)
180 NEXT R
185 REM MOVE SHELL DOWN DIAGONALLY
190 FOR R=1 TO 14
200 C=R+13
210 CALL HCHAR(R,C,142)
215 REM PRODUCE DIFFERENT SHELL NOISE
220 CALL SOUND(20,650,10)
225 REM ERASE SHELL
230 CALL HCHAR(R,C,32)
240 NEXT R
250 GOTO 9000
995 REM DEFINE RISING SHELL
1000 N$="1B3C7EFFFFFFFF" ← 10 Fs
1010 CALL CHAR(141,N$)
1015 REM DEFINE FALLING SHELL
1020 P$="FFFFFFFFF7E3C18"
1030 CALL CHAR(142,P$)
1040 RETURN
9000 CALL CLEAR
RUN
```

Now the shell rises diagonally, turns over when it reaches the top, then falls diagonally, producing a different sound.

We are now ready to add the background scenery—the artillery gun and the target.

We must modify the shell's path to extend from the end of the gun (row 18, column 4) to the top of the target (row 19, column 30).

The loop that makes the shell rise is

```
FOR R=18 TO 6 STEP -1
C=22-R

NEXT R
```

and the one that makes it fall is

```
FOR R=6 TO 19
C=R+11

NEXT R
```

Program 19-4 Artillery Gun Shelling a Factory

```
NEW
100 REM PROGRAM 19-4
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM DRAW ARTILLERY GUN
130 GOSUB 2000
135 REM DRAW TARGET (FACTORY)
140 GOSUB 3000
145 REM MOVE SHELL UP DIAGONALLY
150 FOR R=18 TO 6 STEP -1
160 C=22-R
170 CALL HCHAR(R,C,141)
175 REM PRODUCE SHELL NOISE
180 CALL SOUND(20,450,10)
185 REM ERASE SHELL
190 CALL HCHAR(R,C,32)
200 NEXT R
205 REM MOVE SHELL DOWN DIAGONALLY
210 FOR R=6 TO 19
220 C=R+11
230 CALL HCHAR(R,C,142)
235 REM PRODUCE DIFFERENT SHELL NOISE
240 CALL SOUND(20,650,10)
245 REM ERASE SHELL
250 CALL HCHAR(R,C,32)
260 NEXT R
270 GOTO 9000
995 REM DEFINE ARTILLERY GUN
1000 A$="000000000000307FF" ← 11 zeros
1010 B$="00030F3FFFFCF0C0"
1020 C$="FCFCFC0C000000000" ← 9 zeros
1030 D$="FFFFFFF183C3C18" ← 8 Fs
1035 REM DEFINE FACTORY (TARGET)
1040 E$="01010101030F1F3F"
1050 F$="81818181FFFFFFF" ← 8 Fs
1060 G$="80808080C0F0F8FC"
1070 H$="7FFFC0C0C0C0C0C0"
1080 I$="FFFF000082418241"
1090 J$="FFF30303A353A353"
1100 K$="CAC0C0C0C0C0C0C0"
1110 L$="820000383838FFFF"
1120 M$="A30303030303FFFF"
1125 REM DEFINE TWO SHELLS
1130 N$="183C7EFFFFFFFFF" ← 10 Fs
1140 P$="FFFFFFFFF7E3C18" ← 10 Fs
```

```

1150 CALL CHAR(128,A$)
1160 CALL CHAR(129,B$)
1170 CALL CHAR(130,C$)
1180 CALL CHAR(131,D$)
1190 CALL CHAR(132,E$)
1200 CALL CHAR(133,F$)
1210 CALL CHAR(134,G$)
1220 CALL CHAR(135,H$)
1230 CALL CHAR(136,I$)
1240 CALL CHAR(137,J$)
1250 CALL CHAR(138,K$)
1260 CALL CHAR(139,L$)
1270 CALL CHAR(140,M$)
1280 CALL CHAR(141,N$)
1290 CALL CHAR(142,P$)
1300 RETURN
1995 REM DRAW ARTILLERY GUN
2000 CALL HCHAR(19,1,128)
2010 CALL HCHAR(19,2,129)
2020 CALL HCHAR(19,3,130)
2030 CALL HCHAR(20,1,131,3)
2040 RETURN
2995 REM DRAW FACTORY (TARGET)
3000 CALL HCHAR(18,29,132)
3010 CALL HCHAR(18,30,133)
3020 CALL HCHAR(18,31,134)
3030 CALL HCHAR(19,29,135)
3040 CALL HCHAR(19,30,136)
3050 CALL HCHAR(19,31,137)
3060 CALL HCHAR(20,29,138)
3070 CALL HCHAR(20,30,139)
3080 CALL HCHAR(20,31,140)
3090 RETURN
9000 CALL CLEAR
RUN

```

Finally, we can create an explosion in which the factory collapses and debris flies up in the air.

Program 19-5 Artillery Gun Shelling a Factory, with Explosion

Do not type NEW. Leave Program 19-4 still in memory.

```

100 REM PROGRAM 19-5
265 REM PRODUCE EXPLOSION
270 GOSUB 4000
280 GOTO 9000

3995 REM PRODUCE EXPLOSION
3998 REM ERASE AND MOVE LEFT SIDE OF FACTORY
4000 CALL VCHAR(18,29,32,2) ← Note this is VCHAR!
4010 CALL HCHAR(17,27,132)
4020 CALL HCHAR(16,26,135)
4025 REM PRODUCE SOUND WHEN SHELL HITS
4030 CALL SOUND(1000,-6,10)
4035 REM COLLAPSE RIGHT SIDE OF FACTORY
4040 CALL HCHAR(18,31,92)
4045 REM COLLAPSE MIDDLE OF FACTORY, THROW UP DEBRIS
4050 CALL HCHAR(19,29,35,3)
4060 CALL HCHAR(17,29,42,3)
4065 REM COLLAPSE BOTTOM OF FACTORY, THROW UP DEBRIS
4070 CALL HCHAR(20,29,126,3)
4080 CALL HCHAR(16,28,38,5)

```



```

4085 REM PRODUCE MORE EXPLOSION NOISE
4090 CALL SOUND(1000,-5,12)
4100 CALL SOUND(1000,-7,10)
4105 REM ERASE DEBRIS AS IT SETTLES
4110 CALL HCHAR(16,26,32,7)
4120 CALL HCHAR(17,27,32,6)
4130 RETURN
RUN

```

When creating a complex scene like the explosion in Program 19-5, a paper sketch showing the row and column locations of each character helps considerably. Remember, composing an elaborate scene takes care and patience.

As you create scenes, be sure to save each program on tape or disk; you will be able to use the same characters, movements, or sounds in other programs. Always start with a simple scene. If you cannot get that to look right, you surely will not succeed with a more complex scene.

One way to check a scene is to insert a line like

```
352 GOTO 352
```

This statement branches to itself, creating an endless loop; you must press FCTN and 4 (CLEAR) to regain control. Either STOP or END would actually terminate the program and return control to you. Unfortunately, both make the computer print a message that either distorts the picture or destroys part of it.

One unrealistic aspect of Program 19-5 is that the shell's velocity is constant. In the real world, of course, gravity slows the shell down as it rises and speeds it up as it falls. We can model this effect by inserting a do-nothing loop and changing its length as the shell moves. We will change the loop's upper limit (L) as follows:

1. Line 123 makes L initially 3.
2. Line 184 (inside the upward loop) adds 2 to L, slowing the shell down as it rises.
3. Line 232 (inside the downward loop) subtracts 2 from L, speeding the shell up as it falls.

The new lines are

```

122 REM INITIALIZE SHELL SPEED CONTROL
123 L=3

181 REM REDUCE SHELL SPEED AS IT MOVES UP
182 FOR K=1 TO L
183 NEXT K
184 L=L+2

231 REM INCREASE SPEED AS IT MOVES DOWN
232 L=L-2
233 FOR K=1 TO L
234 NEXT K

```

Let us now combine programs to create another elaborate scene. Here we will command a jet fighter to shoot a missile at a hovering spaceship. The scene looks like this:

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid G

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid H

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid I

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The codes for the jet fighter are

A	B	C	D	E	F
0 0	0 0	0 0	C F	F F	B 0
0 0	0 0	0 0	E F	F F	F C
0 0	0 0	0 0	F F	F F	F E
F F	0 0	0 0	F F	F F	F F
F F	E 0	0 0	E 7	F F	F E
0 F	F F	B 0	C 3	F F	F C
0 F	F B	B 0	0 0	3 B	0 0
0 F	F B	B 0	0 0	3 B	0 0

The codes for the missile are

G	H	I
0 6	F F	C 0
0 7	F F	F B
0 6	F F	C 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0

The codes for the spaceship (as in Chapter 17) are

```
J$="3F408EBE403F0909"
K$="FF003C3C00FF2424"
L$="FC02717102FE9090"
```

Let's first just make the fighter fire the missile.

Program 19-6 Jet Fighter Attacking

```
NEW
100 REM PROGRAM 19-6
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM DRAW JET IN ROW 10, COLUMN 5
130 GOSUB 2000
135 REM DRAW MISSILE BELOW JET
140 C=5
150 GOSUB 3000
155 REM PRESS F KEY TO FIRE MISSILE
160 CALL KEY(0,CH,PR)
170 IF CH<>"F" THEN 160 ← 70 is F.
180 FOR C=5 TO 28
185 REM DRAW MISSILE
190 GOSUB 3000
195 REM MAKE SOME NOISE
200 CALL SOUND(60,-2,10)
205 REM ERASE MISSILE
210 CALL HCHAR(12,C,32,3)
220 NEXT C
230 GOTO 9000

995 REM DEFINE JET FIGHTER
1000 A$="000000FFFF0F0F0F" ← 6 zeros
1010 B$="000000000E0FFBFB" ← 8 zeros
1020 C$="0000000000B0B0B0" ← 10 zeros
1030 D$="CFEEEEFFFE7C30000"
1040 E$="FFFFFFFFFFFF3B3B" ← 12 Fs
1050 F$="B0CFEEEEFFFC0000"
1055 REM DEFINE MISSILE
1060 G$="0607060000000000" ← 10 zeros
1070 H$="FFFFFF0000000000" ← 6 Fs, 10 zeros
1080 I$="C0FB000000000000" ← 11 zeros
1090 CALL CHAR(128,A$)
1100 CALL CHAR(129,B$)
1110 CALL CHAR(130,C$)
1120 CALL CHAR(131,D$)
1130 CALL CHAR(132,E$)
1140 CALL CHAR(133,F$)
1150 CALL CHAR(134,G$)
1160 CALL CHAR(135,H$)
1170 CALL CHAR(136,I$)
1180 RETURN
1995 REM DRAW JET FIGHTER
2000 CALL HCHAR(10,5,128)
2010 CALL HCHAR(10,6,129)
2020 CALL HCHAR(10,7,130)
2030 CALL HCHAR(11,5,131)
2040 CALL HCHAR(11,6,132)
2050 CALL HCHAR(11,7,133)
2060 RETURN
2995 REM DRAW MISSILE
3000 CALL HCHAR(12,C,134)
3010 CALL HCHAR(12,C+1,135)
3020 CALL HCHAR(12,C+2,136)
3030 RETURN
9000 CALL CLEAR
RUN
```

We already know how to move an object up and down (from Program 13-10). Now let's add the alien ship to Program 19-6.

Program 19-7 Jet Fighter Attacking Spaceship

Do not type NEW. Leave Program 19-6 in the computer.

```
100 REM PROGRAM 19-7
155 REM MOVE SPACESHIP DOWN
160 FOR R=1 TO 20
170 GOSUB 4000
175 REM PRESS F KEY TO FIRE
180 CALL KEY(0,CH,PR)
190 IF CH<>"F" THEN 210 ← 70 is F.
200 GOSUB 5000
205 REM ERASE SPACESHIP
210 CALL CHAR(R,27,32,3)
220 NEXT R
225 REM MOVE SPACESHIP UP
230 FOR R=20 TO 1 STEP -1
240 GOSUB 4000
245 REM PRESS F KEY TO FIRE
250 CALL KEY(0,CH,PR)
260 IF CH<>"F" THEN 280 ← 70 is F.
270 GOSUB 5000
275 REM ERASE SPACESHIP
280 CALL CHAR(R,27,32,3)
290 NEXT R
300 GOTO 140
1175 REM DEFINE SPACESHIP
1180 J$="3F408E8E403F0909"
1190 K$="FF003C3C00FF2424"
1200 L$="FC02717102FE9090"
1210 CALL CHAR(137,J$)
1220 CALL CHAR(138,K$)
1230 CALL CHAR(139,L$)
1240 RETURN
3995 REM DRAW SPACESHIP IN 27TH COLUMN
4000 CALL HCHAR(R,27,137)
4010 CALL HCHAR(R,28,138)
4020 CALL HCHAR(R,29,139)
4030 RETURN
4995 REM MOVE MISSILE RIGHT
5000 FOR C=5 TO 27
5005 REM DRAW MISSILE
5010 GOSUB 3000
5015 REM MAKE NOISE
5020 CALL SOUND(60,-2,10)
5025 REM ERASE MISSILE
5030 CALL CHAR(12,C,32,3)
5040 NEXT C
5050 RETURN
9000 CALL CLEAR
RUN
```

See how we put

```
CALL KEY(0,CH,PR)
IF CH<>"F" THEN
GOSUB 5000
```

inside the loops that move the spaceship up and down? The computer uses these statements to check whether the player has pressed the F key. If so, the computer goes to line 5000 and draws the missile moving toward the spaceship. If not, the computer keeps mov-

ing the ship up and down. Note also how nicely the ship recovers even when a missile destroys it completely. The fighter, on the other hand, must wait until the ship reaches the top before it “grows” another missile.

Rather than destroy the target, we can change line 5000 to

```
5000 FOR C=5 TO 26
```

and print the message

```
HIT
```

when a missile hits the spaceship. A hit occurs if the missile reaches column 27 (the end of the loop) when the target is in row 12. We can also produce the sound of an explosion if this happens. Add the lines

```
5045 REM TEST FOR TARGET IN ROW 12
5050 IF R<>12 THEN 5120
5055 REM MAKE EXPLOSION SOUND WHEN MISSILE STRIKES SHIP
5060 CALL SOUND(1000,-5,12)
5065 DISPLAY LETTERS H, I, T
5070 CALL HCHAR(12,27,72)
5080 CALL HCHAR(12,28,73)
5090 CALL HCHAR(12,29,84)
5095 REM CONTINUE EXPLOSION SOUND
5100 CALL SOUND(1000,-7,10)
5105 REM ERASE MESSAGE
5110 CALL HCHAR(12,27,32,3)
5120 RETURN
```

Use Table 6-1 to find the codes for H, I, T.

Try to hit the target five times in a row.

You could put the spaceship in a random column (not too close to the jet fighter—that would make it too easy to hit) by inserting

```
151 REM SELECT RANDOM COLUMN FOR SPACESHIP
152 CA=INT(RND*13)+15 ← Pick a number from 15 to 27.
```

You must then change the lines that draw and erase the spaceship, move the missile, and print and erase the HIT message:

```
210 CALL HCHAR(R,CA,32,3)
280 CALL HCHAR(R,CA,32,3)
4000 CALL HCHAR(R,CA,137)
4010 CALL HCHAR(R,CA+1,138)
4020 CALL HCHAR(R,CA+2,139)
5000 FOR C=5 TO CA-1
5070 CALL HCHAR(12,CA,72)
5080 CALL HCHAR(12,CA+1,73)
5090 CALL HCHAR(12,CA+2,84)
5110 CALL HCHAR(12,CA,32,3)
```

INT(RND*13) picks a random number between 0 and 12, since RND is always less than 1. Adding 15 shifts the range to between 15 and 27.

The loops that move the spaceship up and down are obviously highly repetitive. It would save a lot of typing if we could combine them into a single loop that would make R increase and then decrease. We can do this by using the computer's built-in ABS (*absolute value*) function. The revised section is

```

155 REM MOVE SPACESHIP UP AND DOWN
160 FOR S=1 TO 39
165 REM ROW NUMBER GOES UP, THEN DOWN
170 R=20-ABS(20-S)
180 GOSUB 4000
185 REM PRESS F KEY TO FIRE
190 CALL KEY(0,CH,PR)
200 IF CH<>70 THEN 220
210 GOSUB 5000
215 REM ERASE SPACESHIP
220 CALL HCHAR(R,27,32,3)
230 NEXT S

```

ABS gives the size (*magnitude*) of a number, regardless of its sign. For example, $ABS(3) = ABS(-3) = 3$. So, line 170 works as follows:

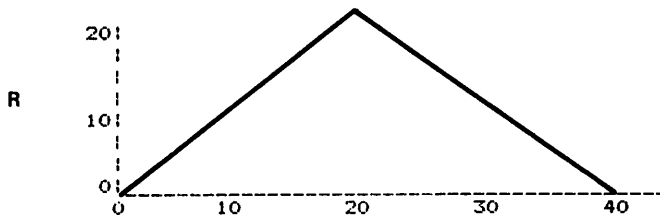
1. As S goes from 1 to 20, $20-S$ is always positive, so $R = 20-ABS(20-S) = 20-(20-S) = S$. Thus R also goes from 1 to 20, and the spaceship moves down the screen.
2. As S goes from 21 to 39, $20-S$ is always negative, so $R = 20-ABS(20-S) = 20-(S-20) = 40-S$. Thus R goes from 19 (when $S=21$) to 1 (when $S=39$), and the spaceship moves up the screen

Note the following graph of R versus S; R goes up and then down, just the way we wanted. Another equivalent sequence is

```

170 R=S
172 REM REVERSE R DIRECTION TO MOVE SPACESHIP UP
174 IF S<21 THEN 180
176 R=40-S

```



20

SHOOTING AT A MOVING TARGET

One obvious flaw in games like Program 19-7 is that the target stops when we fire at it. While this is a noble sacrifice on the target's part, it detracts from the challenge. The problem is that the computer stops moving the target when it starts moving the projectile. We will now develop a more interesting game in which both the target and the projectile move. Let us start by moving a simple target (O) up and down. We will use ABS (absolute value) to make the row number increase and then decrease, as discussed after Program 19-7.

Program 20-1 Simple Target Moving Up and Down

```
NEW
100 REM PROGRAM 20-1
110 CALL CLEAR
115 REM MOVE TARGET DOWN AND UP
120 FOR S=1 TO 19
125 REM MAKE ROW NUMBER GO UP, THEN DOWN
130 R=18-ABS(10-S)
135 REM DRAW TARGET
140 CALL HCHAR(R,20,79)
150 CALL SOUND(50,30000,20)
155 REM ERASE TARGET
160 CALL HCHAR(R,20,32)
170 NEXT S
180 CALL CLEAR
RUN
```

To make the target rise and fall continuously, change line 180 to

```
180 GOTO 120
```

Now you must press FCTN and 4 (CLEAR) to stop the program.

Next we will draw a tank. It will fire a bullet when the player presses the F key. The tank looks like this:

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Its codes are

A	B	C	D	E
3 F	F B	0 0	F F	F B
7 F	F C	0 0	F F	F 0
F B	F D	0 0	F F	F C
F B	F F	F F	2 0	4 0
F B	F F	F F	7 0	E 0
F F	F F	F F	F 9	F 0
F F	F D	0 0	7 0	E 0
F F	F C	0 0	2 0	4 0

The corresponding BASIC statements are

```

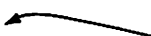
A$="3F7FFBFBFBFFFFF" ← 6 Fs
B$="FBFCFDFFFFFFDFC"
C$="000000FFFFFF0000" ← 6 zeros, 6 Fs, 4 zeros
D$="FFFFFF2070F97020"
E$="FBF0F040E0F0E040"

```

The bullet the tank fires is



and is obtained by

F\$="0000000018000000"  8 zeros, 1, 8, 6 zeros

Program 20-2 Tank Firing a Bullet

```
NEW
100 REM PROGRAM 20-2
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM DRAW TANK
130 GOSUB 2000
135 REM WAIT FOR PLAYER TO PRESS F KEY
140 CALL KEY(0,CH,PR)
150 IF CH<>"F" THEN 140 ← 70 is F.
155 REM FIRE BULLET
160 FOR C=8 TO 20
170 CALL HCHAR(10,C,133)
180 CALL SOUND(100,-2,20)
185 REM ERASE BULLET
190 CALL HCHAR(10,C,32)
200 NEXT C
210 GOTO 9000
995 REM DEFINE TANK
1000 A$="3F7FFBFBFBFFFFF" ← 6 Fs
1010 B$="FBFCFDFFFFFFDFC" ← 7 Fs
1020 C$="000000FFFFFF0000" ← 6 zeros, 6 Fs, 4 zeros
1030 D$="FFFFFF2070F97020" ← 6 Fs
1040 E$="FBF0F040E0F0E040"
1045 REM DEFINE BULLET
1050 F$="0000000018000000" ← 8 zeros, 1, 8, 6 zeros
1060 CALL CHAR(128,A$)
1070 CALL CHAR(129,B$)
1080 CALL CHAR(130,C$)
1090 CALL CHAR(131,D$)
1100 CALL CHAR(132,E$)
1110 CALL CHAR(133,F$)
1120 RETURN
1995 REM DRAW TANK
2000 CALL HCHAR(10,5,128)
2010 CALL HCHAR(10,6,129)
2020 CALL HCHAR(10,7,130)
2030 CALL HCHAR(11,5,131)
2040 CALL HCHAR(11,6,132)
2050 RETURN
9000 CALL CLEAR
RUN
```

Now we must combine Programs 20-1 and 20-2 and make the target and bullet move simultaneously. The easiest way to do this is to give up the convenient FOR-NEXT loop that moves the bullet. Instead, we will explicitly add 1 to C (moving the bullet right one column) each time the target moves up or down a line. That is, within the loop that moves the target, we must start the bullet and move it until it reaches column 20.

But how does the computer know if a bullet has been fired? We certainly don't want it moving a bullet that does not exist. Let us introduce B as an indicator. B is 0 (zero) if no bullet is on the screen and 1 (one) if one is. The program must set B to the following values:

1. 0 initially.
2. 1 when the player presses the F key.
3. Back to 0 when the bullet reaches column 20.

Let us first test these concepts with a program that moves the bullet in a loop controlled by S. We must start the bullet at the end of the gun (column 8) and stop it when it reaches column 20.

Program 20-3 Tank Firing a Bullet at a Target

Do not type NEW. Leave program 20-2 in the computer.

```
100 REM PROGRAM 20-3
135 REM NO BULLET INITIALLY
140 B=0
145 REM START BULLET AT END OF GUN
150 C=8
155 REM CONTROL BULLET IN AN UNRELATED LOOP
160 FOR S=1 TO 19
165 REM START BULLET IF PLAYER PRESSES F KEY
170 CALL KEY(0,CH,PR)
180 IF CH<>"F" THEN 200
185 REM SET B TO INDICATE BULLET FIRED
190 B=1
195 MOVE BULLET IF STILL ON SCREEN
200 IF B<>1 THEN 220
205 REM MOVE BULLET
210 GOSUB 3000
220 NEXT S
230 GOTO 160

2995 REM MOVE BULLET ONE COLUMN
2998 REM DRAW BULLET
3000 CALL HCHAR(10,C,133)
3010 CALL SOUND(100,-2,20)
3020 CALL HCHAR(10,C,32) ← Erases shadow
3030 C=C+1
3035 REM STOP DRAWING BULLET BEYOND TARGET
3040 IF C<20 THEN 3070
3045 REM RETURN BULLET TO END OF BARREL
3050 C=8
3055 REM BULLET NO LONGER ON SCREEN
3060 B=0
3070 RETURN
RUN
```

Program 20-3 moves the bullet in a loop that could also move the target up or down. The game program then combines Programs 20-1 and 20-3.

Program 20-4 Target Practice with a Moving Target

Do not type NEW. Leave Program 20-3 in the computer.

```
100 REM PROGRAM 20-4
165 REM ROW NUMBER INCREASES, THEN DECREASES
170 R=15-ABS(10-S)
175 REM DRAW TARGET
180 CALL HCHAR(R,20,79)
190 CALL SOUND(50,50000,10)
195 REM ERASE TARGET
200 CALL HCHAR(R,20,32)
205 REM START BULLET IF PLAYER PRESSES F KEY
210 CALL KEY(0,CH,PR)
220 IF CH<>"F" THEN 240
225 REM SET B TO INDICATE BULLET FIRED
230 B=1
235 REM MOVE BULLET IF STILL ON SCREEN
240 IF B=1 THEN 260
245 REM MOVE BULLET
250 GOSUB 3000
260 NEXT S
270 GOTO 160
RUN
```

We will now change the target to the spaceship we used in Chapter 17 and in Program 19-7. Its characters are

```
G$="3F408383403F0909"
H$="FF003C3C00FF2424"
I$="FC02717102FE9090"
```

Program 20-5 Target Practice with a Moving Alien Ship

Do not type NEW. Leave Program 20-4 in the computer.

```
100 REM PROGRAM 20-5
180 GOSUB 4000
200 CALL HCHAR(R,20,32,3)

1115 REM DEFINE SPACESHIP
1120 G$="3F408383403F0909"
1130 H$="FF003C3C00FF2424"
1140 I$="FC02717102FE9090"
1150 CALL CHAR(134,G$)
1160 CALL CHAR(135,H$)
1170 CALL CHAR(136,I$)
1180 RETURN

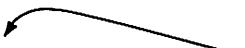
3995 REM DRAW SPACESHIP
4000 CALL HCHAR(R,20,134)
4010 CALL HCHAR(R,21,135)
4020 CALL HCHAR(R,22,136)
4030 RETURN
RUN
```

The bullet is rather slow, since it is moving at the same speed as the spaceship. To make it move at twice the spaceship's speed, simply double its step each time by changing line 3030 to

```
3030 C=C+2
```

You could similarly make the bullet move three or four times as fast as the spaceship.

Rather than shoot a bullet, the tank could fire a laser blast at the spaceship. We will make the laser blast out of a checkerboard pattern; that is


Z\$="000000AA55AA0000" 6 zeros

Program 20-6 Target Practice with a Laser Blast

Do not type NEW. Leave Program 20-5 in memory.

```
100 REM PROGRAM 20-6
```

Note that some REMs from the combined programs no longer apply.

```
1072 Z$="000000AA55AA0000" ← 6 zeros
```

```
1056 CALL CHAR(137,Z$)
```

```
2998 REM DRAW ONE SEGMENT OF LASER BLAST
```

```
3000 CALL HCHAR(10,C,137)
```

```
3010 C=C+1
```

```
3015 REM STOP DRAWING BLAST BEYOND TARGET
```

```
3020 IF C<20 THEN 3060
```

```
3025 REM AT TARGET, ERASE ENTIRE BLAST
```

```
3030 CALL HCHAR(10,8,32,13)
```

```
3035 REM RETURN TO END OF BARREL
```

```
3040 C=8
```

```
3045 REM BLAST NO LONGER ON SCREEN
```

```
3050 B=0
```

```
3060 RETURN
```

```
RUN
```

Like the bullet, the laser blast is slow. You can imagine the tank commander reporting that the blast would be deadly if the target would only wait for it. To double the blast's speed, change lines 3000 and 3010 to

```
3000 CALL HCHAR(10,C,137,2)
```

```
3010 C=C+2
```

We could similarly triple or quadruple the blast's speed, although it may still look more like a snake's tongue unrolling in sections than a deadly laser.

To tell when the player has scored a hit, insert lines that determine whether the blast passed close enough to the target to cause damage. If it does, announce the hit and make an appropriate noise. The additions are

```
3012 REM CHECK IF BLAST WITHIN RANGE  
3013 REM CHECK IF TARGET IS ON BLAST'S PATH  
3014 IF R<>10 THEN 3020
```

```
3015 REM CHECK IF BLAST IS CLOSE ENOUGH TO CAUSE DAMAGE
3016 IF C<18 THEN 3020
3017 GOSUB 6000

5995 REM DECLARE A HIT
6000 CALL HCHAR(10,20,72)
6010 CALL HCHAR(10,21,73)
6020 CALL HCHAR(10,22,84)
6025 REM PRODUCE EXPLOSION NOISE
6030 CALL SOUND(200,-5,12)
6040 CALL SOUND(500,-7,10)
6050 CALL HCHAR(10,20,32,3)
6060 RETURN
```

21

MOVING TWO THINGS SIMULTANEOUSLY

Let us now create a scene in which two objects move in different directions simultaneously. We will have a ship travel right on the surface while a submarine travels left underneath it. This requires sets of CALL HCHARs, one for the ship and the other for the submarine. The ship looks like this:

Grid A								Grid B								Grid B								Grid C							
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

The characters are

```
A$="0000FFB0B6B6403F"  
B$="3C3CFF001C1C00FF"  
C$="0000FF01616204FB"
```

The submarine looks like this:

Grid D								Grid E								Grid F							
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

Its characters are

D\$="000000001F7FFF3F" ← 8 zeros
 E\$="0C0C0C0CFFFFFFF" ← 8 Fs
 F\$="00000000FCFEFFFE" ← 8 zeros

We will start by moving the ship and submarine without any background.

Program 21-1 Ship and Submarine Moving in Different Directions

```

NEW
100 REM PROGRAM 21-1
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM MOVE SHIP AND SUBMARINE
130 FOR C=1 TO 25
135 REM DRAW SHIP
140 GOSUB 2000
145 REM DRAW SUBMARINE
150 GOSUB 3000
155 REM ERASE SHIP
160 CALL HCHAR(9,C,32,4)
165 REM ERASE SUBMARINE
170 CALL HCHAR(17,26+C,32,3)
180 NEXT C
190 GOTO 9000
995 REM DEFINE SHIP
1000 A$="0000FFB08686403F"
1010 B$="0C3CFF001C1C00FF"
1020 C$="0000FF01516204FB"
1025 REM DEFINE SUBMARINE
1030 D$="000000001F7FFF3F"
1040 E$="0C0C0C0CFFFFFFF"
1050 F$="00000000FCFEFFFE"
1055 REM DEFINE CHARACTERS
1060 CALL CHAR(128,A$)

```



```

1070 CALL CHAR(129,B#)
1080 CALL CHAR(130,C#)
1090 CALL CHAR(131,D#)
1100 CALL CHAR(132,E#)
1110 CALL CHAR(133,F#)
1120 RETURN
1995 REM DRAW SHIP
2000 CALL HCHAR(9,C,128)
2010 CALL HCHAR(9,C+1,129,2) ←————— 2 center grids are the same.
2020 CALL HCHAR(9,C+3,130)
2030 RETURN
2995 REM DRAW SUBMARINE
3000 CALL HCHAR(17,28-C,131)
3010 CALL HCHAR(17,29-C,132)
3020 CALL HCHAR(17,30-C,133)
3030 RETURN
9000 CALL CLEAR
RUN

```

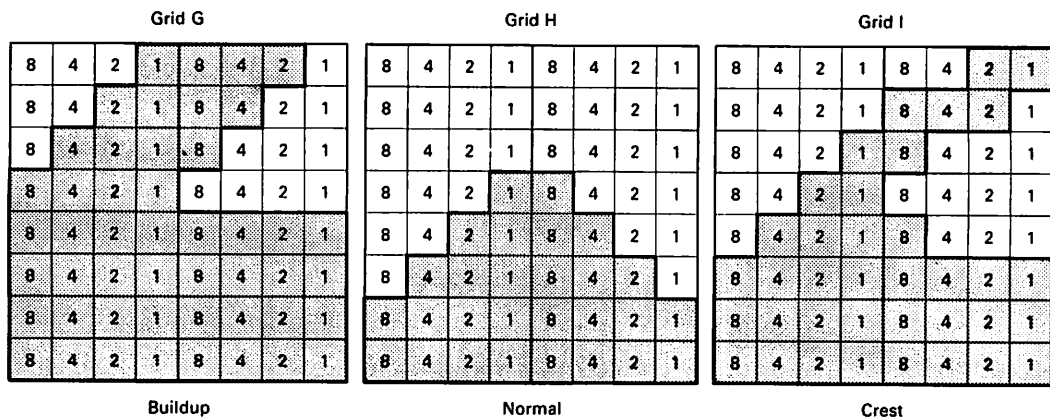
You can slow the motion down with either CALL SOUND statements or do-nothing loops.

We can draw rolling ocean waves by superimposing the following three pictures:

Wave #1

#2

#3



giving

```

G#="1E3C78F0FFFFFFFF" ←————— 8 Fs
H#="0000001B3C7EFFFF"
I#="030E1B3078FFFFFF" ←————— 6 Fs

```

We can also draw birds with flapping wings as follows

Grid J								Grid K							
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

giving

```
J$="0000187EDB818100"
K$="8181DB7E18000000" ← 6 zeros
```

Program 21-2 Ship, Submarine, Ocean Waves, and Birds

Do not enter NEW. Leave Program 21-1 in memory.

```
151 REM DRAW BIRDS AND OCEAN WAVES
152 GOSUB 4000

1115 REM DEFINE BIRDS AND OCEAN WAVES
1120 G$="1E3C78FOFFFFFFFF" ← 8 Fs
1130 H$="000000183CEFFFF" ← 6 zeros
1140 I$="030E183078FFFFFF" ← 6 Fs
1150 J$="0000187EDB818100"
1160 K$="8181DB7E18000000" ← 6 zeros
1170 CALL CHAR(134,G$)
1180 CALL CHAR(135,H$)
1190 CALL CHAR(136,I$)
1200 CALL CHAR(137,J$)
1210 CALL CHAR(138,K$)
1220 RETURN

3995 REM DRAW BIRDS AND OCEAN WAVES
3998 REM WAVES BUILDING UP
4000 CALL HCHAR(10,1,134,31)
4005 REM BIRDS WITH WINGS IN ONE POSITION
4010 CALL HCHAR(2,5,137)
4020 CALL HCHAR(2,20,137)
4025 REM WAVES CRESTING
4030 CALL HCHAR(10,1,136,31)
4035 REM BIRDS WITH WINGS IN ANOTHER POSITION
4040 CALL HCHAR(2,5,138)
4050 CALL HCHAR(2,20,138)
4055 REM WAVES NORMAL
4060 CALL HCHAR(10,1,135,31)
4070 RETURN
RUN
```

In creating such a complex picture, minor typing errors, especially in character definitions, may lead to strange results. You may want to introduce endless loops into your program to check each individual picture by itself. For example, to check the first wave picture, add

```
4015 GOTO 4015
```

Before continuing, you must press FCTN and 4 (CLEAR) to stop the program and remove the loop (that is, type 305, then ENTER).

Let us now make the submarine fire a missile when the ship reaches column 10.

The pictures of the ship, submarine, birds, and waves do not change. We must add a new picture for the launch sequence and make the computer draw it only when C=10. We will start the launch sequence in line 5000. The only addition to the main program is

```
153 REM LAUNCH MISSILE WHEN SHIP REACHES COLUMN 10
154 IF C<> 10 THEN 160
156 GOSUB 5000
```

The missile looks like this:

Grid L

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

and its character is

```
L$="1818181818183C7E" ← 6 18s
```

Program 21-3 Submarine Firing a Missile

Do not type NEW since we want to retain Program 21-2.

```
100 REM PROGRAM 21-3
153 REM LAUNCH MISSILE WHEN SHIP REACHES COLUMN 10
154 IF C<>10 THEN 160
155 GOSUB 5000

1215 REM DEFINE THE MISSILE
1220 L$="18181818183C7E" ← 6 18s
1230 CALL CHAR(139,L$)
1240 RETURN

4995 REM LAUNCH MISSILE
5000 FOR R=16 TO 1 STEP -1
5010 CALL HCHAR(R,28-C,139)
5015 REM PRODUCE SOUND
5020 CALL SOUND(20,330,10)
5025 REM ERASE MISSILE
5030 CALL HCHAR(R,28-C,32)
5040 NEXT R
5050 RETURN
RUN
```

The submarine now fires a missile when C is 10. Note how the action stops, since the computer is no longer executing the parts of the program that redraw and thus move the birds, ship, waves, and submarine. The missile breaks through the ocean waves (leaving a hole in them) and flies off the screen.

We can easily let the player control when the submarine fires a missile by using the F key, as in earlier programs. The additional lines are

```
153 REM LAUNCH MISSILE WHEN PLAYER PRESSES F KEY
154 CALL KEY(0,CH,PR)
155 IF CH<>70 THEN 160
157 GOSUB 5000
```

Now you can make the submarine fire several missiles. You do not have to rush because nothing moves during the launch. See if you can score a direct hit on a bird. Be careful; don't practice this feat in front of Audubon Society members. Fortunately, birds, the ocean, and even the ship recuperate quickly from the effects of a missile and soon continue their normal movements.

The complete program with the player controlling the firing is as follows:

Program 21-4 Ocean, Birds, Ship, and Submarine Firing Missiles

```
NEW
100 REM PROGRAM 21-4
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM MOVE SHIP AND SUBMARINE
130 FOR C=1 TO 25
135 REM DRAW SHIP
140 GOSUB 2000
145 REM DRAW SUBMARINE
150 GOSUB 3000
```

```

151 REM DRAW BIRDS AND OCEAN WAVES
152 GOSUB 4000
153 REM LAUNCH MISSILE WHEN PLAYER PRESSES F KEY
154 CALL KEY(0,CH,PR)
155 IF CH<>"F" THEN 160
156 GOSUB 5000
157 REM ERASE SHIP
158 CALL HCHAR(9,C,32,4)
159 REM ERASE SUBMARINE
160 CALL HCHAR(17,28-C,32,3)
161 NEXT C
162 GOTO 9000

995 REM DEFINE SHIP
1000 A$="0000FF80B6B6403F"
1010 B$="3C3CFF001C1C00FF"
1020 C$="0000FF01616204F8"
1025 REM DEFINE SUBMARINE
1030 D$="000000001F7FFF3F" ← 8 zeros
1040 E$="0C0C0C0CFFFFFFF" ← 8 Fs
1050 F$="00000000FCFEFFFE" ← 8 zeros
1055 REM DEFINE CHARACTERS
1060 CALL CHAR(128,A$)
1070 CALL CHAR(129,B$)
1080 CALL CHAR(130,C$)
1090 CALL CHAR(131,D$)
1100 CALL CHAR(132,E$)
1110 CALL CHAR(133,F$)
1115 REM DEFINE OCEAN WAVES
1120 G$="1E3C7BF0FFFFFFF" ← 8 Fs
1130 H$="0000001B3C7EFFFF" ← 6 zeros
1140 I$="030E1B307BFFFFFF" ← 6 Fs
1145 REM DEFINE BIRDS
1150 J$="00001B7EDB018100"
1160 K$="8181DB7E1B000000" ← 6 zeros
1170 CALL CHAR(134,G$)
1180 CALL CHAR(135,H$)
1190 CALL CHAR(136,I$)
1200 CALL CHAR(137,J$)
1210 CALL CHAR(138,K$)
1215 REM DEFINE THE MISSILE
1220 L$="1B1B1B1B1B1B3C7E" ← 6 1Bs
1230 CALL CHAR(139,L$)
1240 RETURN
1995 REM DRAW SHIP
2000 CALL HCHAR(9,C,128)
2010 CALL HCHAR(9,C+1,129,2) ← 2 center grids are the same.
2020 CALL HCHAR(9,C+3,130)
2030 RETURN
2995 REM DRAW SUBMARINE
3000 CALL HCHAR(17,28-C,131)
3010 CALL HCHAR(17,29-C,132)
3020 CALL HCHAR(17,30-C,133)
3030 RETURN
3995 REM DRAW BIRDS AND OCEAN WAVES
3998 REM WAVES BUILDING UP
4000 CALL HCHAR(10,1,134,31)
4005 REM BIRDS WITH WINGS IN ONE POSITION
4010 CALL HCHAR(2,5,137)
4020 CALL HCHAR(2,20,137)
4025 REM WAVES CRESTING
4030 CALL HCHAR(10,1,136,31)
4035 REM BIRDS WITH WINGS IN ANOTHER POSITION
4040 CALL HCHAR(2,5,138)
4050 CALL HCHAR(2,20,138)
4055 REM WAVES NORMAL
4060 CALL HCHAR(10,1,135,31)
4070 RETURN

```

```

4995 REM LAUNCH MISSILE
5000 FOR R=16 TO 1 STEP -1
5005 REM DRAW MISSILE
5010 CALL HCHAR(R,28-C,139)
5015 REM PRODUCE SOUND ,
5020 CALL SOUND(20,330,10)
5025 REM ERASE MISSILE
5030 CALL HCHAR(R,28-C,32)
5040 NEXT R
5050 RETURN
9000 CALL CLEAR
RUN

```

As with the shell in Chapter 19, gravity should reduce the missile's speed. We can simulate gravity by increasing the duration of the missile's sound in line 5020. Let us call the duration D, start D at 20, and increase it by 10 after each iteration. The required changes are

```

4998 REM START DURATION AT 20
5000 D=20
5005 FOR R=16 TO 1 STEP -1

5020 CALL SOUND(D,330,10)
5021 REM SLOW MISSILE BY MAKING SOUND LONGER
5022 D=D+10

```

An additional complication here is that the missile moves from water into air. Clearly, water should exert more drag than air, since water is much thicker (more viscous). We can simulate that difference by reducing the change in D after the missile reaches the surface ($R=10$). To do this, we replace line 5022 with

```

5022 D=D+5
5023 REM SLOW MISSILE FURTHER IN WATER
5024 IF R<10 THEN 5030
5025 D=D+5

```

Now D increases by 10 while the missile is in water but only by 5 when it reaches air.

22

RACE CAR GAME

We can use a solid square (16 Fs) to draw a simple rectangular racetrack for a car. The top and bottom are in rows 9 and 17, respectively, while the sides are in columns 9 and 24. We start the drawing instructions on line 3000 so that we can use them later as a subroutine in a racing game.

Program 22-1 Beginner's Racetrack

```
NEW
100 REM PROGRAM 22-1
105 REM DEFINE SOLID SQUARE
110 SQ$="FFFFFFFFFFFFFFFF" ← 16 Fs
120 CALL CHAR(148,SQ$)
130 CALL CLEAR
2995 REM DRAW RACETRACK
2998 REM DRAW TOP AND BOTTOM
3000 CALL HCHAR(9,9,148,16) ←
3020 CALL HCHAR(17,9,148,16) ← HCHARs for horizontal lines
3025 REM DRAW SIDES
3030 CALL VCHAR(9,9,148,9) ←
3040 CALL VCHAR(9,24,148,9) ← VCHARs for vertical lines
RUN
```

To race a car around the track, we need the following pictures of it moving right, left, up, or down.

Moving Left

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Moving Right

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid G

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid H

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Moving Up

Grid I

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid J

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid K

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid L

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Moving Down

Grid M

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid N

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid P

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid Q

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The custom characters are as follows:

Moving Left

```
A$="103E3E103F7FFFFFF"
B$="3B7C7C3BFCFFFFFF"
C$="111F7F3F103E3E10"
D$="FCFFFFFFC3B7C7C3B"
```

Moving Right

```
E$="103E3E103F7FFFFFF"
F$="3B7C7C3BFCFFFFFF"
G$="3FFF7F3F103E3E10"
H$="1FFFFFFC3B7C7C3B"
```

Moving Up

```
I$="0007FFFFFF6F0F"
J$="000F6FFFFFFF6F0"
K$="0F6F1FFFFFF6F06F"
L$="F0F6FFFFFF6F06F"
7 Fs
```

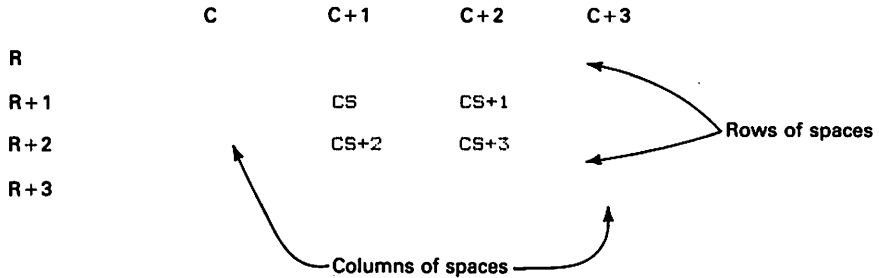
Moving Down

```
M$="06066FFFFFFF6F0F"  
N$="60606FFFFFFF6F0"  
P$="0F6FFFFFFF6F3703"  
Q$="F0F6FFFFFF6E0C0"
```



7 Fs

We can draw a generalized car just like the generalized starship in Chapter 18. That is, each car looks like this:



The only difference in the four pictures is the starting character CS: CS=128 for the car moving left, 132 for right, 136 for up, and 140 for down.

We will use the same control keys as in Program 18-5 (the starship game with momentum), that is,

E (up arrow)	Move car up.
S (left arrow)	Move car left.
D (right arrow)	Move car right.
X (down arrow)	Move car down.
Q	Quit (end the game).

Like the starship in Program 18-5, the car continues in its current direction until a player presses a different direction key. The car starts moving left from row 1, column 10 (top center).

Program 22-2 Car Racing Around a Track

```
NEW  
100 REM PROGRAM 22-2  
105 REM DEFINE ALL CHARACTERS  
110 GOSUB 2000  
120 CALL CLEAR  
125 REM DRAW RACETRACK  
130 GOSUB 3000  
135 REM START CAR IN ROW 1, COLUMN 10  
140 R=1  
150 C=10  
155 REM CAR MOVES LEFT INITIALLY  
160 CS=128  
170 DI$="L"  
180 GOSUB 1000
```

```

185 REM WAIT FOR KEYBOARD COMMANDS
190 CALL KEY(O,CH,PR)
195 REM Q (QUIT) KEY ENDS GAME
200 IF CH=81 THEN 9000 ← 81 is Q.
205 REM LEFT ARROW (S KEY) MAKES DIRECTION LEFT
210 IF CH<>83 THEN 250 ← 83 is S (left-arrow).
220 DI$="L"
230 CS=128
240 GOTO 360
245 REM RIGHT ARROW (D KEY) MAKES DIRECTION RIGHT
250 IF CH<>68 THEN 290 ← 68 is D (right-arrow).
260 DI$="R"
270 CS=132
280 GOTO 360
285 REM UP ARROW (E KEY) MAKES DIRECTION UP
290 IF CH<>69 THEN 330 ← 69 is E (up-arrow).
300 DI$="U"
310 CS=136
320 GOTO 360
325 REM DOWN ARROW (X KEY) MAKES DIRECTION DOWN
330 IF CH<>88 THEN 360 ← 88 is X (down-arrow).
340 DI$="D"
350 CS=140
355 REM MOVE CAR IN DIRECTION DI$
358 REM MOVE CAR LEFT
360 IF DI$<>"L" THEN 390
365 REM KEEP CAR ON SCREEN LEFT
370 IF C=1 THEN 390
380 C=C-1
390 IF DI$<>"R" THEN 420
395 REM KEEP CAR ON SCREEN RIGHT
400 IF C=29 THEN 420
410 C=C+1
415 REM MOVE CAR UP
420 IF DI$<>"U" THEN 450
425 REM KEEP CAR ON SCREEN UP
430 IF R=1 THEN 450
440 R=R-1

445 REM MOVE CAR DOWN
450 IF DI$<>"D" THEN 480
455 REM KEEP CAR ON SCREEN DOWN
460 IF R=21 THEN 480
470 R=R+1
480 GOSUB 1000
490 GOTO 190
995 REM GENERALIZED CAR PICTURE
998 REM PUT SPACES AROUND PICTURE
1000 CALL HCHAR(R,C+1,32,2) ← 2 HCHARs
1010 CALL HCHAR(R+3,C+1,32,2) ← 2 HCHARs
1020 CALL VCHAR(R+1,C,32,2) ← 2 VCHARs
1030 CALL VCHAR(R+1,C+3,32,2) ← 2 VCHARs
1035 REM PUT CAR IN CENTER (CS IS STARTING CHARACTER)
1040 CALL HCHAR(R+1,C+1,CS)
1050 CALL HCHAR(R+1,C+2,CS+1)
1060 CALL HCHAR(R+2,C+1,CS+2)
1070 CALL HCHAR(R+2,C+2,CS+3)
1080 RETURN
1995 REM DEFINE ALL CHARACTERS
1998 REM RACE CAR MOVING LEFT
2000 A$="1C3E3E1C3F7FFFFF"
2010 B$="387C7C38FCFFFFFC"
2020 C$="FFFF/F3F1C3E3E1C"
2030 D$="FCFFFFFC387C7C38"
2035 REM RACE CAR MOVING RIGHT
2040 E$="1C3E3E1C3FFFFF3F"

```

```

2050 F$="387C7C38FCFEFFFF"
2060 G$="3FFFFFF3F1C3E3E1C"
2070 H$="FFFFFFEFC387C7C38"
2075 REM RACE CAR MOVING UP
2080 I$="03076FFFFFFFF6F0F" ←
2090 J$="C0E0F6FFFFFFFF6F0" ←
2100 K$="0F6FFFFFFFF6F0606" ←
2110 L$="F0F6FFFFFFFF66060" ←
2115 REM RACE CAR MOVING DOWN
2120 M$="06066FFFFFFFF6F0F" ←
2130 N$="6060F6FFFFFFFF6F0" ←
2140 P$="0F6FFFFFFFF6F3703" ←
2150 Q$="F0F6FFFFFFFF6E0C0" ←
2155 REM DEFINE CAR CHARACTERS
2160 CALL CHAR(128,A$)
2170 CALL CHAR(129,B$)
2180 CALL CHAR(130,C$)
2190 CALL CHAR(131,D$)
2195 REM RACE CAR MOVING RIGHT
2200 CALL CHAR(132,E$)
2210 CALL CHAR(133,F$)
2220 CALL CHAR(134,G$)
2230 CALL CHAR(135,H$)
2235 REM RACE CAR MOVING UP
2240 CALL CHAR(136,I$)
2250 CALL CHAR(137,J$)
2260 CALL CHAR(138,K$)
2270 CALL CHAR(139,L$)

2275 REM RACE CAR MOVING DOWN
2280 CALL CHAR(140,M$)
2290 CALL CHAR(141,N$)
2300 CALL CHAR(142,P$)
2310 CALL CHAR(143,Q$)
2315 REM RACE TRACK SYMBOL
2320 SQ$="FFFFFFFFFFFFFFFF" ← 16 Fs
2330 CALL CHAR(148,SQ$)
2340 RETURN
2995 REM DRAW RACETRACK
2998 REM DRAW TOP AND BOTTOM
3000 CALL HCHAR(9,9,148,16)
3010 CALL HCHAR(17,9,148,16)
3015 REM DRAW SIDES
3020 CALL VCHAR(9,9,148,9)
3030 CALL VCHAR(9,24,148,9)
3040 RETURN
9000 CALL CLEAR
RUN

```

Type RUN and test your program by driving the car around the track. Don't crash into the walls. Leave yourself lots of room; since the car has spaces all around it, it can destroy a wall just by coming near it.

Program 22-2 is quite long, but you can easily obtain it by modifying Program 18-4. All you must do is change the character definitions, add the racetrack-drawing routine, and adjust a few lines in the main program.

We can easily draw a more complex racetrack that requires more skillful maneuvering. All we must change is the track-drawing subroutine. The new version is as follows:

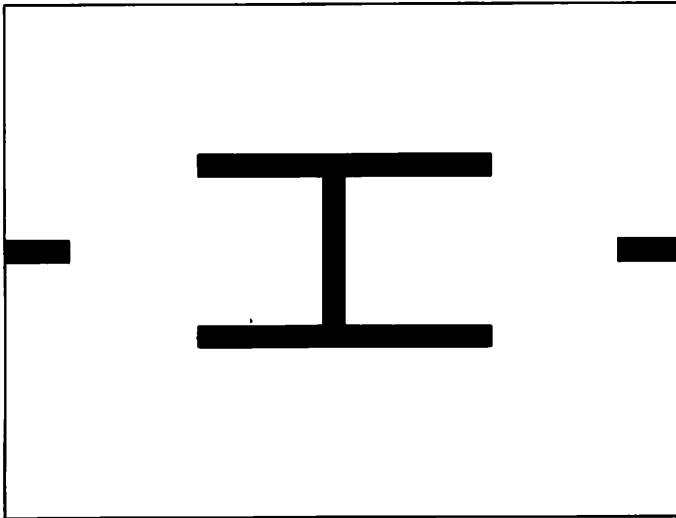
Program 22-3 Intermediate Racing Game

Do not type NEW. Leave Program 22-2 in memory.

```
100 REM PROGRAM 22-3

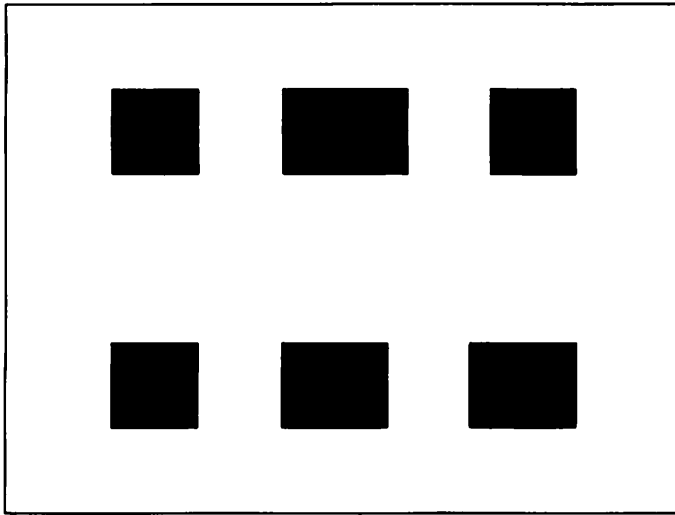
2995 REM DRAW INTERMEDIATE RACING TRACK
2998 REM DRAW TOP AND BOTTOM
3000 CALL HCHAR(8,10,148,14)
3010 CALL HCHAR(16,10,148,14)
3015 REM DRAW MIDDLE AND 2 OUTSIDE BOUNDARIES
3020 CALL VCHAR(9,16,148,7)
3030 CALL HCHAR(12,1,148,3)
3040 CALL HCHAR(12,30,148,3)
3050 RETURN
RUN
```

The screen will look like this:

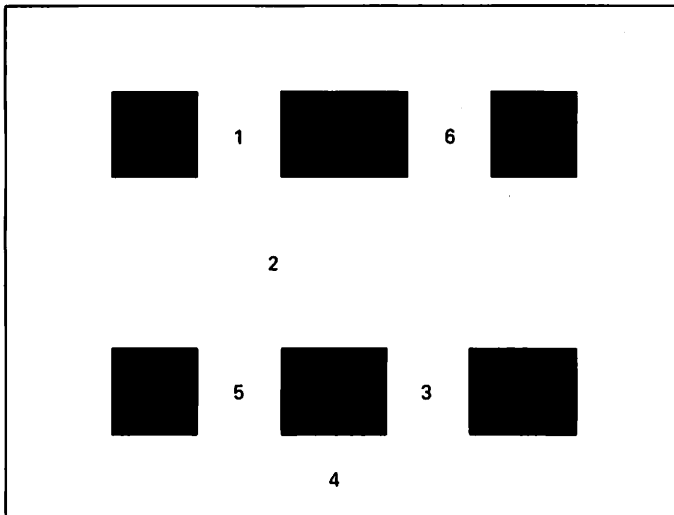


Try racing the car around this track. Tricky, isn't it? Remember that the car has momentum, so you have to prepare for each turn by putting your finger on an arrow key before the car reaches an opening. If the car passes an opening, you do have the luxury of being able to turn it around and go back. Real race-car drivers do not have this option—imagine how many head-on collisions would occur if they did!

A more complex racetrack has several openings. Since the car with spaces around it is four lines tall and four columns wide, all openings must be at least that large for the car to pass through. The advanced track looks like this:



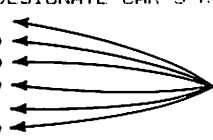
To indicate the route the car is to take, we can place numbers on the track. The track then looks like this:



Program 22-4 Advanced Racetrack

```
NEW
100 REM PROGRAM 22-4

2995 REM DRAW ADVANCED RACE TRACK
2998 REM DRAW TOP BARRIERS
3000 FOR R=5 TO 9
3010 CALL HCHAR(R,6,148,4)
3020 CALL HCHAR(R,14,148,6)
3030 CALL HCHAR(R,24,148,4)
3040 NEXT R
3045 REM DRAW BOTTOM BARRIERS
3050 FOR R=17 TO 20
3060 CALL HCHAR(R,6,148,4)
3070 CALL HCHAR(R,14,148,5)
3080 CALL HCHAR(R,23,148,5)
3090 NEXT R
3095 REM USE NUMBERS TO DESIGNATE CAR'S ROUTE
3100 CALL HCHAR(7,11,49)
3110 CALL HCHAR(12,13,50)
3120 CALL HCHAR(18,11,53)
3130 CALL HCHAR(22,16,52)
3140 CALL HCHAR(7,22,54)
3150 CALL HCHAR(18,21,51)
3160 RETURN
RUN
```



Look up the codes for the digits in Table 6-1.

You can easily devise other geometries and enter them into the track-drawing subroutine. Give the driver a challenge, but also give him or her a reasonable chance to avoid crashing. One approach is to make the track more difficult (say, by lengthening the top and bottom barriers in Program 22-4) each time the driver completes a circuit without crashing. We could also simplify the track each time the driver fails, thus keeping a beginner from getting discouraged. Many arcade games become more difficult as the player becomes more skillful; this practice holds the player's interest and keeps the quarters coming.

23

STARSHIP GAME

We will now develop a game in which the player moves a starship along the bottom of the screen and fires laser blasts at targets. The starship looks like this:

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Its four custom characters are

```
A$="0103070F07010143"  
B$="80C0E0F0E08080C2"  
C$="E3E7EFFFFFE143E7"  
D$="C7E7F7FFFFB7C2E7"
```

We start the starship in column 15 of row 20. As in previous games, we use the arrow keys to move the starship right or left and the F key to fire a laser blast. First, let's draw the starship.

Program 23-1 Stationary Starship

```
NEW  
100 REM PROGRAM 23-1  
105 REM DEFINE ALL CHARACTERS  
110 GOSUB 1000  
120 CALL CLEAR  
125 REM START STARSHIP IN COLUMN 15  
130 C=15  
135 REM DRAW STARSHIP  
140 GOSUB 2000  
150 GOTO 9000  
995 REM DEFINE ALL CHARACTERS  
1000 A$="0103070F07010143"  
1010 B$="80C0E0F0E08080C2"  
1020 C$="E3E7EFFFFFE143E7" ← 5 Fs  
1030 D$="C7E7F7FFFFB7C2E7"  
1040 CALL CHAR(128,A$)  
1050 CALL CHAR(129,B$)  
1060 CALL CHAR(130,C$)  
1070 CALL CHAR(131,D$)  
1080 RETURN  
1995 REM PICTURE OF STARSHIP  
1998 REM SPACES AROUND STARSHIP  
2000 CALL VCHAR(20,C,32,2)  
2010 CALL VCHAR(20,C+3,32,2)  
2015 REM STARSHIP IN CENTER  
2020 CALL HCHAR(20,C+1,128)  
2030 CALL HCHAR(20,C+2,129)  
2040 CALL HCHAR(21,C+1,130)  
2050 CALL HCHAR(21,C+2,131)  
2060 RETURN  
9000 END  
RUN
```

Next we introduce the following control keys:

S (left arrow)	Move starship left.
D (right arrow)	Move starship right.
F	Fire a laser blast.
Q	Quit (end the game).

Because the starship with a space on each side occupies four columns, we must not start drawing it beyond column 29.

Program 23-2 Starship Moving Along the Bottom

```
NEW
100 REM PROGRAM 23-2
105 REM DEFINE ALL CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM PLACE STARSHIP IN COLUMN 15
130 C=15
135 REM DRAW STARSHIP
140 GOSUB 2000
145 REM LET PLAYER MOVE STARSHIP
150 CALL KEY(O,CH,PR)
155 REM Q KEY QUITTS GAME
160 IF CH=B1 THEN 9000 ← 81 is Q.
165 REM LEFT ARROW (S KEY) MOVES STARSHIP LEFT
170 IF CH<>B3 THEN 200 ← 83 is S.
180 IF C=1 THEN 200
190 C=1-1
195 REM RIGHT ARROW (D KEY) MOVES STARSHIP RIGHT
200 IF CH<>68 THEN 230 ← 68 is D.
210 IF C=29 THEN 230
220 C=C+1
225 REM F KEY FIRES LASER
230 IF CH<>70 THEN 140 ← 70 is F.
240 PRINT "FIRE LASER"
250 GOTO 9000
995 REM DEFINE ALL CHARACTERS
1000 A$="0103070F07010143"
1010 B$="80C0E0F0E08080C2"
1020 C$="E3E7EFFFFFE143E7"
1030 D$="C7E7F7FFFF87C2E7"
1040 CALL CHAR(128,A$)
1050 CALL CHAR(129,B$)
1060 CALL CHAR(130,C$)
1070 CALL CHAR(131,D$)
1080 RETURN
1995 REM PICTURE OF STARSHIP
1998 REM SPACES AROUND STARSHIP
2000 CALL VCHAR(20,C,32,2)
2010 CALL VCHAR(20,C+3,32,2)
2015 REM STARSHIP IN CENTER
2020 CALL HCHAR(20,C+1,128)
2030 CALL HCHAR(20,C+2,129)
2040 CALL HCHAR(21,C+1,130)
2050 CALL HCHAR(21,C+2,131)
2060 RETURN
9000 CALL CLEAR
RUN
```

Type RUN. Move the starship to the far right and far left to verify the limits. Now press the F key. You should see the message FIRE LASER. As before, we have a simple program stub as a stand-in for the routine that will create the laser blast.

Now let us introduce the laser ray. It looks like this:

Grid LL								Grid LR							
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

The left and right halves are the characters

LL\$="0201020102010201" ← 4 0201s
 LR\$="8040804080408040" ← 4 8040s

When the spaceship fires, a ray beams up from the bottom of the screen to the top. The program must later print spaces over the ray to erase the trail. The routine that fires the laser and erases it is

```

3995 REM FIRE LASER BLAST
4000 FOR R=19 TO 1 STEP -1
4010 CALL HCHAR(R,C+1,132)
4020 CALL HCHAR(R,C+2,133)
4030 NEXT R
4035 REM ERASE LASER TRAIL
4040 FOR R=19 TO 1 STEP -1
4050 CALL HCHAR(R,C+1,32,2) ← Prints 2 spaces over ray
4060 NEXT R
4070 RETURN
  
```

Let us now put targets near the top of the screen, scattering them randomly in columns 2 through 31 and rows 1 through 5. Remember that there are spaces around the starship, so we can never have a laser blast in column 1.

```

4995 REM PRODUCE 10 RANDOM TARGETS
5000 FOR J=1 TO 10
5005 REM SELECT COLUMN AND ROW RANDOMLY
5010 C=INT(RND*30)+2
5020 R=INT(RND*5)+1
5030 CALL HCHAR(R,C,42) ← 42 is *.
5040 NEXT J
5050 RETURN
  
```

The FOR-NEXT loop produces ten random row and column numbers. Note, however, that some numbers might be duplicates, so there will sometimes be fewer than ten targets. To obtain a different series of targets each time you run the program, add

```
102 RANDOMIZE
```

Program 23-3 Starship Firing a Laser Blast at Targets

Do not type NEW. Leave Program 23-2 in memory.

```
100 REM PROGRAM 23-3
102 RANDOMIZE

121 REM CREATE TARGETS
122 GOSUB 5000

235 REM DRAW AND ERASE LASER BLAST
240 GOSUB 4000
250 GOTO 140

1075 REM DEFINE LASER BLAST
1080 LL$="0201020102010201" ← 4 0201s
1090 LR$="8040804080408040" ← 4 8040s
1100 CALL CHAR(132,LL$)
1110 CALL CHAR(133,LR$)
1120 RETURN

3995 REM FIRE LASER BLAST
4000 FOR R=19 TO 1 STEP -1
4010 CALL HCHAR(R,C+1,132)
4020 CALL HCHAR(R,C+2,133)
4030 NEXT R
4035 REM ERASE LASER TRAIL
4040 FOR R=19 TO 1 STEP -1
4050 CALL HCHAR(R,C+1,32,2) ← Prints 2 spaces over ray
4060 NEXT R
4070 RETURN

4995 REM PRODUCE 10 RANDOM TARGETS
5000 FOR J=1 TO 10
5005 REM SELECT COLUMN AND ROW RANDOMLY
5010 C=INT(RND*30)+2
5020 R=INT(RND*5)+1
5030 CALL HCHAR(R,C,42) ← 42 is *.
5040 NEXT J
5050 RETURN
RUN
```

Hitting the stationary targets is simple. For a greater challenge, use a stopwatch to see how fast you can destroy all of them.

One problem is that it takes a long time to move the starship across the screen. We could introduce another control key (W) that sends the ship into a random hyperspace warp; that is, pressing W moves the ship to a random column. If the next target is far away, you may want to try a warp rather than just moving the ship one column at a time.

The additions are

```
230 IF CHK>70 THEN 260 ← 70 is F.  
255 REM W KEY SENDS STARSHIP INTO HYPERSPACE WARP  
260 IF CHK>87 THEN 140 ← 87 is W.  
265 REM HYPERSPACE WARP - ERASE STARSHIP  
270 CALL HCHAR(20,C+1,32,2)  
280 CALL HCHAR(21,C+1,32,2)  
285 REM CHOOSE NEW POSITION RANDOMLY  
290 C=INT(RND*29)+1  
300 GOTO 140
```

To create fancier targets and increase the adventure involved in shooting at them, we can use the bomb drawing from Chapter 18. The revised target looks like this:

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

and the custom character is

```
BOMB$="2442BD3C3CBD4224"
```

We will also make one target (the last one) trigger a doomsday machine. When you hit it with a laser blast, the explosion vaporizes the entire scene.

The revisions to the target-generating routine are

```
5030 CALL HCHAR(R,C,134)  
5045 REM SAVE COORDINATES OF LAST TARGET AS DOOMSDAY TRIGGER  
5050 CN=C  
5060 RN=R  
5070 RETURN
```

Program 23-4 Starship Attacking Targets, Including Doomsday

Do not type NEW. Leave Program 23-3 in memory.

```
100 PROGRAM 23-4
103 REM END OF THE WORLD INDICATOR
104 EDW=0

241 REM CHECK IF WORLD IS ENDING
242 IF EDW=0 THEN 140
243 REM EXPLOSION MARKS END OF WORLD
244 CALL SOUND(1000,-5,10)
245 CALL SOUND(1000,-7,10)
246 CALL HCHAR(1,1,42,768)
247 PRINT "SORRY - WORLD JUST ENDED"
248 PRINT "BETTER LUCK NEXT GAME"
249 GOTO 9000

1115 REM DEFINE BOMB
1120 BOMB$="2442BD3C3CBD4224"
1130 CALL CHAR(134,BOMB$)
1140 RETURN

4024 REM ARE WE IN NEUTRON BOMB'S ROW?
4025 IF R=RN THEN 4030
4026 REM DID WE HIT THE NEUTRON BOMB?
4027 IF CN=C+1 THEN 4080
4028 IF CN=C+2 THEN 4080

4075 REM HIT DOOMSDAY TRIGGER, SET END OF WORLD INDICATOR
4080 EDW=1
4090 RETURN
RUN
```

The aim here is to destroy as many targets as you can as fast as possible before you blow up the entire world. Note that you cannot tell which target triggers the doomsday machine. Mastery of this game qualifies you for a career in the top echelons of world diplomacy.

24

PLANETARY ROVER GAME

Rather than shoot at targets, we could dodge meteorites falling toward a planetary rover. As before, we will use the arrow keys (S and D) to move the rover left or right. To make the game more challenging, we will restrict the rover's movement. The planetary rover looks like this:

Grid A

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid B

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid C

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid D

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid E

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid F

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The custom characters are

```

A$="F0FOFFFFFF3331F"
B$="3C3CFFFFFF1818FF" ← 6 Fs
C$="0FOFFFFFFCCBF8" ← 7 Fs
D$="3F7FFF86B6B686FF"
E$="FFFFFF18DB18FF"
F$="FCFEFF616D6D1FF"

```

To start, we will limit the rover's range to columns 9 (CLOW) through 22 (CHIGH). Let us first draw the vehicle and move it across the screen. Note that the picture occupies five columns (counting the spaces on each side) and is two lines high.

Program 24-1 Planetary Rover Moving Right and Left

```

NEW
100 REM PROGRAM 24-1
105 REM DEFINE CUSTOM CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM DEFINE ROVER'S LIMITS, STARTING COLUMN
130 CLOW=9
140 C=15
150 CHIGH=22
155 REM DRAW GROUND AND BOUNDARIES
160 GOSUB 3000
165 REM DRAW ROVER
170 GOSUB 2000
175 REM CHECK KEYBOARD FOR COMMANDS
180 CALL KEY(0,CH,PR)
185 REM Q KEY IS QUIT (END GAME)
190 IF CH=B1 THEN 9000 ← 81 is Q
195 REM LOOK FOR MOVEMENT COMMANDS
200 GOSUB 4000
210 GOTO 180
995 REM DEFINE CUSTOM CHARACTERS
1000 A$="F0FOFFFFFF33331F" ← 6 Fs
1010 B$="3C3CFFFFFF1818FF"
1020 C$="0FOFFFFFFFCC8BF8" ← 7 Fs
1030 D$="3F7FFF86B6B6B6FF"
1040 E$="FFFFFF18DBDB18FF" ← 6 Fs
1050 F$="FCFEFF616D6D61FF"
1060 CALL CHAR(128,A$)
1070 CALL CHAR(129,B$)
1080 CALL CHAR(130,C$)
1090 CALL CHAR(131,D$)
1100 CALL CHAR(132,E$)
1110 CALL CHAR(133,F$)
1120 RETURN
1995 REM PICTURE OF ROVER
1998 REM PUT SPACES AROUND ROVER
2000 CALL VCHAR(20,C,32,2)
2010 CALL VCHAR(20,C+4,32,2)
2015 REM DRAW ROVER IN MIDDLE
2020 FOR J=1 TO 3
2030 CALL HCHAR(20,C+J,127+J)
2040 CALL HCHAR(21,C+J,130+J)
2050 NEXT J
2060 RETURN
2995 REM DRAW GROUND AND BOUNDARIES
2998 REM DRAW GROUND
3000 CALL HCHAR(22,1,45,31)
3005 REM DRAW BOUNDARIES
3010 CALL VCHAR(1,CLOW-1,124,21)
3020 CALL VCHAR(1,CHIGH+1,124,21)
3030 RETURN

3995 REM LOOK FOR DIRECTION KEYS
3998 REM LEFT-ARROW (S KEY) MOVES ROVER LEFT
4000 IF CH<>B3 THEN 4030
4010 IF C=CLOW THEN 4030
4020 C=C-1
4025 REM RIGHT ARROW (D KEY) MOVES ROVER RIGHT
4030 IF CH<>6B THEN 4060
4035 REM KEEP ROVER WITHIN RANGE RIGHT
4040 IF C=CHIGH-4 THEN 4060 ← Rover with 3 spaces is 5 columns
                                wide, so we cannot start it
                                beyond CHIGH-4.
4050 C=C+1
4060 RETURN
9000 CALL CLEAR
RUN

```

Move the rover right and left. The boundaries act like stone walls; the rover simply refuses to move any farther in its current direction when it reaches one of them.

Now let us introduce falling meteorites. We will represent them as diamond shapes that look like this:

Grid MTR

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The custom character is

```
MTR$="1B3C7EFFFF7E3C1B"
```

The instructions to select a random column between CLOW and CHIGH are

```
RANGE=CHIGH-CLOW+1
CMET1=INT(RND*RANGE)+CLOW
```

RND produces a number between 0 and 1; it can be 0, but it can never be exactly 1. The smallest value of CMET1 occurs when RND=0; that value is

```
CMET1=INT(0)+CLOW=CLOW
```

The largest value occurs when RND is almost 1. Then $\text{INT}(\text{RND} \times \text{RANGE}) = \text{RANGE} - 1$, since RND is always less than 1 and INT drops the fractional part of a number. The value is

```
CMET1=RANGE-1+CLOW=CHIGH-CLOW+1-1+CLOW=CHIGH
```

Thus the meteorite is always between CLOW and CHIGH inclusive (that is, it could also be in either CLOW or CHIGH).

Program 24-2 Planetary Rover with Falling Meteorite

Do not type NEW. Leave Program 24-1 in memory.

```
100 REM PROGRAM 24-2

165 REM CREATE METEORITE IN RANDOM COLUMN
170 RANGE=HIGH-CLOW+1
180 CMET1=INT(RND*RANGE)+CLOW
190 FOR R=1 TO 18
195 REM DRAW ROVER
200 GOSUB 2000
205 REM CHECK KEYBOARD FOR COMMAND
210 CALL KEY(0,CH,PR)
215 REM Q KEY IS QUIT(END GAME)
220 IF CH=81 THEN 9000 ← 81 is Q.
225 REM LOOK FOR MOVEMENT COMMANDS
230 GOSUB 4000
235 REM ERASE OLD METEORITE
240 CALL HCHAR(R,CMET1,32)
245 REM DRAW NEW METEORITE
250 CALL HCHAR(R+1,CMET1,134)
260 NEXT R
265 REM ERASE FINAL PICTURE OF METEORITE
270 CALL HCHAR(19,CMET1,32) ← Loop never erases last picture.
280 GOTO 170

1115 REM DEFINE METEORITE
1120 MTR$="183C7EFFFF7E3C18"
1130 CALL CHAR(134,MTR$)
1140 RETURN
RUN
```

Press the arrow keys to move the rover left and right and dodge the meteorite. Not much of a challenge, is it? You can make the action a bit more exciting by doubling the meteorite's speed. Change lines 190 and 250 as follows:

```
190 FOR R=1 TO 17 STEP 2
250 CALL HCHAR(R+2,CMET1,134)
```

Still, dodging is easy because you have lots of room in which to move the rover, and the meteorite is only one column wide. To make the game more challenging, have two meteorites fall simultaneously, and let them both be two columns wide instead of just one. We must now generate two random numbers using

```
RANGE=HIGH-CLOW+1
CMET1=INT(RND*RANGE)+CLOW
CMET2=INT(RND*RANGE)+CLOW
```

The largest column number must be CHIGH-1 because a meteorite now occupies two columns. The changes in the program are

```

170 RANGE=CHIGH-CLOW+1
180 CMET1=INT(RND*RANGE)+CLOW
182 CMET2=INT(RND*RANGE)+CLOW
240 CALL HCHAR(R,CMET1,32,2)
242 CALL HCHAR(R,CMET2,32,2)
250 CALL HCHAR(R+1,CMET1,134,2)
252 CALL HCHAR(R+1,CMET2,134,2)
270 CALL HCHAR(20,CMET1,32,2)
272 CALL HCHAR(20,CMET2,32,2)

```

When a meteorite strikes the rover, we can stop the computer and report the hit. A hit occurs when the meteorite ends its fall in a column occupied by the rover (that is, in columns C+1 through C+3, excluding the spaces we added to avoid shadows). For example, meteorite 1 hits the rover if

1. $CMET1+1 = C+1$ (left edge of meteorite hits right edge of rover).
2. $CMET1 = C+1$ through $C+2$ (entire meteorite hits rover).
3. $CMET1 = C+3$ (right edge of meteorite hits left edge of rover).

We may summarize these possibilities by saying that a hit occurs if CMET1 lies between C and C+3 inclusive. So the IF-THEN statements to check for a hit are

```

4990 REM CHECK FOR HIT
4995 REM CHECK IF METEORITE 1 STRIKES ROVER
5000 IF CMET1<C THEN 5020
5010 IF CMET1<C+4 THEN 5040
5015 REM CHECK IF METEORITE 2 STRIKES ROVER
5020 IF CMET2<C THEN 5110
5030 IF CMET2>C+3 THEN 5110
5035 REM HIT - MAKE EXPLOSION SOUND
5040 CALL SOUND(1000,-7,10)
5045 REM FLASH HIT MESSAGE
5050 FOR J=1 TO 10
5060 CALL HCHAR(24,15,72)
5070 CALL HCHAR(24,16,73)
5080 CALL HCHAR(24,17,84)
5085 REM ERASE "HIT" TO PRODUCE FLASHING
5090 CALL HCHAR(24,15,32,3)
5100 NEXT J
5110 RETURN

```

Use Table 6-1 to find the codes for H, I, T.

We must also add

```

275 REM CHECK FOR HIT
280 GOSUB 5000
290 GOTO 170

```

Program 24-3 Planetary Rover Game

Do not type NEW. Leave Program 24-2 in memory.

```
170 RANGE=CHIGH-CLOW+1
180 CMET1=INT(RND*RANGE)+CLOW
182 CMET2=INT(RND*RANGE)+CLOW
240 CALL HCHAR(R,CMET1,32,2)
242 CALL HCHAR(R,CMET2,32,2)
250 CALL HCHAR(R+1,CMET1,134,2)
252 CALL HCHAR(R+1,CMET2,134,2)
270 CALL HCHAR(20,CMET1,32,2)
272 CALL HCHAR(20,CMET2,32,2)
275 REM CHECK FOR HIT
280 GOSUB 5000
290 GOTO 170

4990 REM CHECK FOR HIT
4995 REM CHECK IF METEORITE 1 STRIKES ROVER
5000 IF CMET1<C THEN 5020
5010 IF CMET1<C+4 THEN 5040
5015 REM CHECK IF METEORITE 2 STRIKES ROVER
5020 IF CMET2<C THEN 5110
5030 IF CMET2>C+3 THEN 5110
5035 REM HIT - MAKE EXPLOSION SOUND
5040 CALL SOUND(1000,-7,10)
5045 REM FLASH HIT MESSAGE
5050 FOR J=1 TO 10
5060 CALL HCHAR(24,15,72)
5070 CALL HCHAR(24,16,73)
5080 CALL HCHAR(24,17,84)
5085 REM ERASE "HIT" TO PRODUCE FLASHING
5090 CALL HCHAR(24,15,32,3)
5100 NEXT J
5110 RETURN
RUN
```

Use Table 6-1 to find the codes for H, I, T.

Sometimes the two meteorites either overlap or merge. This occurs when CMET1 and CMET2 are either the same or differ by only 1. You can keep the meteorites apart by inserting the following statements:

```
183 REM PICK ANOTHER RANDOM NUMBER IF METEORITES NOT SEPARATED
184 IF ABS(CMET2-CMET1)<3 THEN 182
```

As we noted earlier, ABS (absolute value) determines the size of a number, regardless of its sign.

If you still find the game too easy, try adding one or more of the following features:

1. Give the rover momentum so that it continues in its current direction until you press the other arrow key.
2. Keep track of the number of hits, and explode the rover after a specified number. You could also have each hit cause damage and make fragments fly up in the air. Perhaps a direct hit could cause more damage than a glancing blow.
3. Have a meteorite occasionally enter a hyperspace warp (that is, move to a random column). You could allow one warp per descent, or even a random number of warps. You should also give the player a warp command that moves the rover to a random column. This will provide a chance to escape from a meteorite that suddenly appears directly on top of the rover.

25

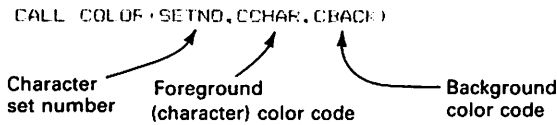
USING COLOR

So far, we have hardly mentioned the fascinating subject of color. Of course, those of you with black-and-white television sets will have to be content with watching variations of gray. The TI has 16 colors; Table 25-1 lists the codes for them.

TABLE 25-1 TI-99/4A COLOR CODES

Color	Color Code
Transparent	1
Black	2
Medium green	3
Light green	4
Dark blue	5
Light blue	6
Dark red	7
Cyan	8
Medium red	9
Light red	10
Dark yellow	11
Light yellow	12
Dark green	13
Magenta	14
Gray	15
White	16

Transparent makes the current screen color show through when the computer displays a character. To change the computer's normal color selection, you must indicate both the character (*foreground*) color and the color of the space around the character (the *background*). You do this with the statement



The TI divides its characters into 12 sets (listed in Appendix A). Note that after the computer executes **CALL COLOR**, all characters in a particular set will appear in the established foreground and background colors. You cannot change the color of a single character because colors are assigned a set at a time.

The computer's normal color choices are:

1. Black characters on a cyan (greenish blue) background when you are entering programs.
2. Black characters on a light green background when the computer is running a program.

To see how each color looks, we can simply make the foreground and background the same. Since the screen normally contains all spaces (a space is code 32, which is in character set 1), we will see a solid screen in a particular color.

Program 25-1 Kaleidoscope of Screen Colors

```

NEW
100 REM PROGRAM 25-1
110 CALL CLEAR
120 FOR CCODE=1 TO 16
130 PRINT "COLOR ";CCODE
140 CALL COLOR(1,CCODE,CCODE)
150 CALL SOUND(2000,30000,10) ← Wait a while.
160 CALL CLEAR
170 NEXT CCODE
RUN
  
```

The code for each color appears at the bottom of the screen. Note that the border is always the usual light green. This obviously makes light green blend with the border and medium green difficult to distinguish from it. Note that transparent disappears quickly, but the other 15 colors linger for about 2 seconds. For people who are not artists, cyan is greenish blue and magenta is reddish purple. You should take this opportunity to make the shading as distinct as possible by adjusting the color on your television set.

Let us now illustrate how characters look in different color combinations. The following program prints ten + symbols (+ is in character set 2) near the center of the screen in various foreground and background colors.

Program 25-2 Kaleidoscope of Colors with Characters

```
NEW
100 REM PROGRAM 25-2
110 FOR CCHAR=1 TO 16
120 FOR CBACK=1 TO 16
130 CALL CLEAR
140 CALL COLOR(2,CCHAR,CBACK)
150 PRINT "CHAR COLOR","BACK COLOR"
160 PRINT CCHAR,CBACK
170 CALL HCHAR(12,10,43,10) ← 43 is +.
180 CALL SOUND(3000,30000,10) ← Wait a while.
190 NEXT CBACK
200 NEXT CCHAR
210 CALL CLEAR
RUN
```

The program displays each possible choice of colors for about 3 seconds; however, since there are 256 combinations, the program will take almost 15 minutes to run. Note the following:

1. The characters disappear when the background and foreground colors are the same. You can think of this as the equivalent of writing in invisible ink.
2. Some combinations make the characters difficult to see (for example, light green characters on a dark green background).
3. You can make the colors more distinct by changing the screen from its normal light green to white. To do this, make character set 1 white on white by inserting

```
105 CALL COLOR(1,16,16)
```

Remember, the space character is in set 1, and CALL CLEAR puts spaces everywhere on the screen.

Now let us try coloring a picture. We will use the moving alien from Program 15-3, which looks like this:

```
\--/
[OO]
/--\
```

Note that / and – are in character set 2, O is in set 6, and [, \, and] are in set 8. To make the entire alien medium red on a white background, we must make the foreground color 9 and the background color 16 for character set 2, 6, and 8.

Program 25-3 Red-and-White Alien Moving Diagonally

```
NEW
100 REM PROGRAM 25-3
105 REM COLORS ARE MEDIUM RED ON WHITE BACKGROUND
110 CALL COLOR(2,9,16)
120 CALL COLOR(8,9,16)
130 CALL COLOR(8,9,16)
140 CALL CLEAR
150 FOR J=1 TO 22
155 REM DRAW ALIEN
160 GOSUB 1000
165 REM PRODUCE SOUND
170 CALL SOUND(20,-3,15)
180 CALL SOUND(5,125,15)
185 REM ERASE ALIEN
190 GOSUB 2000
200 NEXT J
210 GOTO 9000

990 REM DRAW ALIEN
995 REM DRAW TOP      \--/
1000 CALL HCHAR(J,J,92)
1010 CALL HCHAR(J,J+1,45,2)
1020 CALL HCHAR(J,J+3,47)
1025 REM DRAW MIDDLE [00]
1030 CALL HCHAR(J+1,J,91)
1040 CALL HCHAR(J+1,J+1,79,2)
1050 CALL HCHAR(J+1,J+3,93)
1055 REM DRAW BOTTOM  /--\
1060 CALL HCHAR(J+2,J,47)
1070 CALL HCHAR(J+2,J+1,45,2)
1080 CALL HCHAR(J+2,J+3,92)
1090 RETURN
1990 REM ERASE ALIEN
2000 FOR ROW=J TO J+2
2010 CALL HCHAR(ROW,J,32,4)
2020 NEXT ROW
2030 RETURN
9000 CALL CLEAR
RUN
```

The white background makes the alien look like an X-ray or a cutout picture superimposed on the wrong background. You can avoid this effect by making the alien's background color either light green (4) or transparent (1). However, the red alien is much more difficult to see with either of these backgrounds.

To make the alien's eyes dark blue on a light yellow background, change line 120 to

```
120 CALL COLOR(6,5,12)
```

Note that this changes just the eyes because the Os are the only characters we use from set 6.

You must be careful if your picture contains spaces. Changing the background color of character set 1 changes all the empty parts of the screen. Spaces are, after all, just background color.

To make the alien multicolored, we can assign different colors to each character set. Note, however, that – and / will always be in the same colors, since they are in the same set. This applies also to [, \, and]. For example, we could use

```
110 CALL COLOR(2,3,2) ← Medium green on black for set 2
120 CALL COLOR(6,4,5) ← Light green on dark blue for set 6
130 CALL COLOR(8,2,14) ← Black on magenta for set 8
```

in which case – and / will be medium green on a black background, O will be light green on a dark blue background, and [, \, and] will be black on a magenta background.

But what about custom characters you design yourself? The TI assigns character codes 128 through 159 to sets as follows:

128 through 135 are set 13.
 136 through 143 are set 14.
 144 through 151 are set 15.
 152 through 159 are set 16.

For example, let us define a solid square:

```
SQ$="FFFFFFFFFFFFFF" ← 16 Fs
```

We can draw a solid red line across the middle of a white screen as follows:

Program 25-4 Solid Red Line Across a White Screen

```
NEW
100 REM PROGRAM 25-4
105 REM DEFINE CUSTOM CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM MAKE SCREEN WHITE (SPACE IS IN SET 1)
130 CALL COLOR(1,16,16)
135 REM SOLID RED LINE
140 CALL COLOR(13,9,16)
150 CALL HCHAR(12,1,128,32)
160 GOTO 9000
995 REM DEFINE CUSTOM CHARACTERS
1000 SQ$="FFFFFFFFFFFFFF" ← 16 Fs
1010 CALL CHAR(128,SQ$)
1020 RETURN
9000 END
RUN
```

The problem here is that the computer immediately turns the screen light blue and prints READY, distorting the picture and the color. You can stop this from happening by changing line 160 to

```
160 GOTO 160
```

Of course, you will now have to press FCTN and 4 (CLEAR) to regain control.

By defining the same character in another set, we can draw a solid blue line under the solid red line. Just add

```

155 REM SOLID BLUE LINE
160 CALL COLOR(14,5,16)
170 CALL HCHAR(13,1,136,32)
180 GOTO 180
1020 CALL CHAR(136,SQ#)
1030 RETURN

```

By defining triangular characters, we can draw a red house with a green roof. The new characters are

Grid TR

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid TL

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Program 25-5 Red House with a Green Roof

```
NEW
100 REM PROGRAM 25-5
105 REM DEFINE CUSTOM CHARACTERS
110 GOSUB 1000
120 CALL CLEAR
125 REM MAKE SCREEN WHITE (SPACE IS IN SET 1)
130 CALL COLOR(1,16,16)
135 REM DARK GREEN ROOF
140 CALL COLOR(13,13,16)
145 REM DRAW POINTED TOP
150 CALL HCHAR(8,6,128)
160 CALL HCHAR(8,7,129)
165 REM DRAW REST OF HOUSE
170 CALL HCHAR(9,5,128)
180 CALL HCHAR(9,6,130,2)
190 CALL HCHAR(9,8,129)
195 REM MEDIUM RED HOUSE (BLUE BACKGROUND FOR USE IN FLAG LATER)
200 CALL COLOR(14,9,5)
210 FOR R=10 TO 12
220 CALL HCHAR(R,5,136,4)
230 NEXT R
240 GOTO 240
995 REM DEFINE CUSTOM CHARACTERS
998 REM DEFINE TRIANGLES
1000 TR$="0103070F1F3F7FFF"
1010 TL$="80C0E0F0F8FCFEFF"
1015 REM DEFINE SOLID SQUARE
1020 SQ$="FFFFFFFFFFFFFFFF" ← 16 Fs
1030 CALL CHAR(128,TR$)
1040 CALL CHAR(129,TL$)
1050 CALL CHAR(130,SQ$)
1055 REM DEFINE HOUSE CHARACTER
1060 CALL CHAR(136,SQ$)
1070 RETURN
RUN
```

Note that we must use separate character sets for custom characters that we want to color differently. For this reason, we defined two solid squares, one for drawing the red house and the other for drawing the green roof.

By using the left half, checkerboard, the stripes from Table 16-1, as well as a narrow pole, we can add a door, a window, and a red-and-blue-striped flag.

Program 25-6 Little Red Schoolhouse

Do not type NEW. Leave Program 25-5 in memory.

```
100 PROGRAM 25-6
235 REM BLACK DOOR WITH RED BACKGROUND
240 CALL COLOR(15,2,9)
250 CALL HCHAR(12,6,144)
255 REM BLUE WINDOW
260 CALL COLOR(16,5,16)
270 CALL HCHAR(11,7,152)
275 REM GREEN FLAGPOLE
280 CALL VCHAR(6,6,131,2)
295 REM BLUE AND RED STRIPED FLAG
290 CALL HCHAR(6,7,137)
300 GOTO 300
```

```

1065 REM DEFINE DOOR CHARACTER (LEFT HALF-SQUARE)
1070 LH$="FOFOFOFOFOFOFOFO" ← 8 F0s
1080 CALL CHAR(144,LH$)
1085 REM DEFINE WINDOW CHARACTER (CHECKERBOARD)
1090 CB$="AA55AA55AA55AA55" ← 4 AA55s
1100 CALL CHAR(152,CB$)
1105 REM DEFINE FLAGPOLE CHARACTER (SOLID LINE)
1110 PLE$="O1O1O1O1O1O1O1O1" ← 8 O1s
1120 CALL CHAR(131,PLE$)
1125 REM DEFINE FLAG CHARACTER (STRIPES)
1130 FLA$="FF00FF00FF00FF00" ← 4 FF00s
1140 CALL CHAR(137,FLA$)
1150 RETURN
RUN

```

We can also make objects change color as they move. The following program draws a bouncing ball that changes from yellow to red to green to blue. The ball looks like this:

Grid Ball

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Program 25-7 Bouncing Ball Changing Color

```

NEW
100 REM PROGRAM 25-7
110 CALL CLEAR
115 REM DEFINE CUSTOM BALL CHARACTER
120 BALL$="003C7E7E7E7E3C00"
130 CALL CHAR(128,BALL$)
135 REM MAKE SCREEN WHITE
140 CALL COLOR(1,16,16)
145 REM BOUNCE BALL RIGHT
150 FOR C=1 TO 29 STEP 4
155 REM MAKE BALL YELLOW AT BOTTOM
160 CALL COLOR(13,12,16)
170 CALL HCHAR(24,C,128)
180 CALL SOUND(100,30000,30)
185 REM ERASE YELLOW BALL
190 CALL HCHAR(24,C,32)
195 REM MAKE BALL RED ON WAY UP
200 CALL COLOR(13,9,16)
210 CALL HCHAR(23,C+1,128)
220 CALL SOUND(100,30000,30)

```

```

225 REM ERASE RED BALL
230 CALL HCHAR(23,C+1,32)
235 REM MAKE BALL GREEN AT TOP
240 CALL COLOR(13,3,16)
250 CALL HCHAR(22,C+2,128)
260 CALL SOUND(100,30000,30)
265 REM ERASE GREEN BALL
270 CALL HCHAR(22,C+2,32)
275 REM MAKE BALL BLUE ON WAY DOWN
280 CALL COLOR(13,6,16)
290 CALL HCHAR(23,C+3,128)
300 CALL SOUND(100,30000,30)
305 REM ERASE BLUE BALL
310 CALL HCHAR(23,C+3,32)
320 NEXT C
330 CALL CLEAR
RUN

```

Similarly, by defining diagonals to form rays, we can create a fancy exploding star like the ones often seen in video games. The color changes from light yellow to light red and back as the star explodes outward and collapses. The four pictures of the star are

1) Original

Grid CTR1

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

2) Partially extended

Grid LRD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid LLD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid CTR2

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid ULD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid URD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

3) Fully extended

Grid RS

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid SL

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid CTR3

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid SL

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid RS

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

4) Collapsing

Grid URD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid ULD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid CTR2

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid LLD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid LRD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

The custom characters are

Grid CTR1

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid CTR2

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid CTR3

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid SL

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid RS

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid ULD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid URD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid LLD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Grid LRD

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

Program 25-8 Random Multicolored Exploding Stars

```

NEW
100 REM PROGRAM 25-8
105 REM DEFINE CUSTOM CHARACTERS
110 GOSUB 1000
115 REM ASSIGN COLORS
118 REM MAKE OVERALL BACKGROUND WHITE
120 CALL COLOR(1,16,16)
125 REM MAKE ORIGINAL STAR LIGHT YELLOW
130 CALL COLOR(13,12,16)
135 REM MAKE PARTIALLY EXTENDED AND COLLAPSING STARS LIGHT RED
140 CALL COLOR(14,10,16)
145 REM MAKE FULLY EXTENDED STAR DARK RED
150 CALL COLOR(5,7,16)
160 CALL CLEAR
170 FOR J=1 TO 20
175 REM SELECT A RANDOM STARTING POSITION
180 R=INT(RND*23)+1
190 C=INT(RND*20)+1
195 REM DRAW FIVE SUCCESSIVE STAR PICTURES
200 GOSUB 2000
210 GOSUB 3000
220 GOSUB 4000
230 GOSUB 5000
240 GOSUB 2000

```

```

245 REM ERASE STAR
250 GOSUB 6000
260 NEXT J
270 GOTO 9000
995 REM DEFINE CUSTOM CHARACTERS
998 REM DEFINE CENTER #1 (ORIGINAL STAR)
1000 CTR1$="0000183C3C180000"
1005 REM DEFINE CENTER #2 AND HALF-DIAGONALS
1010 CTR2$="00183C7E7E3C1800"
1015 REM HALF-DIAGONALS
1018 REM UPPER LEFT, UPPER RIGHT, LOWER LEFT, LOWER RIGHT
1020 ULD$="8040201000000000"
1030 URD$="0102040800000000"
1040 LLD$="0000000010204080"
1050 LRD$="0000000008040201"
1055 REM DEFINE CENTER #3 AND DIAGONALS
1060 CTR3$="183C7EFFFF7E3C18"
1065 REM DIAGONALS
1068 REM SL$ IS LOWER LEFT TO UPPER RIGHT
1069 REM RS$ IS UPPER LEFT TO LOWER RIGHT
1070 SL$="0102040810204080"
1080 RS$="8040201008040201"
1085 REM DEFINE CHARACTERS
1090 CALL CHAR(128,CTR1$)
1100 CALL CHAR(136,CTR2$)
1110 CALL CHAR(137,ULD$)
1120 CALL CHAR(138,URD$)
1130 CALL CHAR(139,LLD$)
1140 CALL CHAR(140,LRD$)
1150 CALL CHAR(144,CTR3$)
1160 CALL CHAR(145,SL$)
1170 CALL CHAR(146,RS$)
1180 RETURN
1995 REM ORIGINAL STAR
2000 CALL HCHAR(R,C,32,3)
2010 CALL HCHAR(R+1,C+1,128)
2020 CALL HCHAR(R+2,C,32,3)
2030 RETURN
2995 REM PARTIALLY EXTENDED STAR
3000 CALL HCHAR(R,C,140)
3010 CALL HCHAR(R,C+2,139)
3020 CALL HCHAR(R+1,C+1,136)
3030 CALL HCHAR(R+2,C,138)
3040 CALL HCHAR(R+2,C+2,137)
3050 RETURN
3995 REM FULLY EXTENDED STAR
4000 CALL HCHAR(R,C,146)
4010 CALL HCHAR(R,C+2,145)
4020 CALL HCHAR(R+1,C+1,144)
4030 CALL HCHAR(R+2,C,145)
4040 CALL HCHAR(R+2,C+2,146)
4050 RETURN
4995 REM COLLAPSING STAR
5000 CALL HCHAR(R,C,137)
5010 CALL HCHAR(R,C+2,138)
5020 CALL HCHAR(R+1,C+1,136)
5030 CALL HCHAR(R+2,C,139)
5040 CALL HCHAR(R+2,C+2,140)
5050 RETURN
5995 REM ERASE STAR
6000 FOR ROW=R TO R+2
6010 CALL HCHAR(ROW,C,32,3)
6020 NEXT ROW
6030 RETURN
9000 CALL CLEAR
RUN

```

We can also extend our random art program (Program 10-3) to include color. We assign a different color to each character set, starting with dark blue for set 1. The background is always light green.

Program 25-9 Random Art with Color

```
NEW
100 REM PROGRAM 25-9
110 CALL CLEAR
115 REM ASSIGN DIFFERENT COLORS TO CHARACTER SETS
120 FOR SETNO=1 TO 12
130 CALL COLOR(SETNO,SETNO+4,4)
140 NEXT SETNO
150 FOR J=1 TO 100
155 REM SELECT RANDOM ROW, COLUMN
160 R=INT(RND*24)+1
170 C=INT(RND*32)+1
175 REM SELECT RANDOM CHARACTERS
180 CH=INT(RND*95)+33
190 CALL HCHAR(R,C,CH)
200 NEXT J
210 GOTO 150
RUN
```

The program will run forever, producing random characters in random colors. To make character set 1 black and white, insert

```
110 CALL COLOR(1,2,15)
120 FOR SETNO=2 TO 12
```

This makes the screen start out white, since CALL CLEAR puts spaces everywhere. You could also define additional custom characters and produce random art with them. Remember to put the custom characters in different sets so that you can assign them different colors.

APPENDIX

TI COLOR SETS

SET 1

Code	Character
32	(space)
33	!
34	"
35	#
36	\$
37	%
38	&
39	'

SET 2

Code	Character
40	(
41	)
42	*
43	+
44	,
45	—
46	.
47	/

SET 3

Code	Character
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7

SET 4

Code	Character
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

SET 5

Code	Character
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G

SET 6

Code	Character
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O

SET 7

Code	Character
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W

SET 8

Code	Character
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

SET 9

Code	Character
96	'
97	A
98	B
99	C
100	D
101	E
102	F
103	G

SET 10

Code	Character
104	H
105	I
106	J
107	K
108	L
109	M
110	N
111	O

SET 11

Code	Character
112	P
113	Q
114	R
115	S
116	T
117	U
118	V
119	W

SET 12

Code	Character
120	x
121	y
122	z
123	{
124	:
125	}
126	-
127	

INDEX

A

ABS, 181, 220
Advanced car race, 204–5
Advanced race track, 204–5
Alien, 49, 67, 85–86, 90, 104
Alien spacecraft, 25, 96–98, 105
Alphabetic characters, 27, 30, 33, 119
 strings, 27, 30, 39
ALPHA LOCK, 6–7, 11
Amper sign, 123
Animation, 63–69
Applications, 1
Arcade games, 78, 206
Areas on screen, 14
Arithmetic quiz, 42–43
Arrow drawing, 58–59
Arrow keys (cursor control), 42
Art, random, 236
Artillery gun, 166–67
ASC, 47
ASCII, 45
Asking questions, 27–44

B

Background, 221
Backward FOR-NEXT statements, 19,
 53, 112
Backward index, 19, 53, 112
Ball, 59–60, 228
 rolling, 108
BASIC, 8
Birds flying, 192
Birthday card, 33
Blank line, 31
Blank screen, 5
Blanks in drawing, 224
Blastoff, 24–25
Blinking eyes, 66, 84
Bomb, codes for, 99, 143–44
Bomb game, 99–107
Bombing raid, 101
Branches, 42
Bug, 50, 65, 71, 163–66
Bullet, 184
Bunny, 36

C

CALL:

- CHAR, 119
- CLEAR, 22–23, 53
- COLOR, 222
- HCHAR, 45, 48
- KEY, 76, 101
- SOUND, 81–86
- VCHAR, 51
- Capital letters, 10
 - random, 74
- Car, 55, 92, 113, 197–206
- Cassette recorder, 3, 18
- Cat, 34
- Central Processing Unit, 3
- Character code table, 46
- Characters, 4, 46, 118
 - custom, 117–18
- Christmas card, 32
- Christmas tree, 32
- CLEAR, 10, 16, 50
- Clear screen, 11, 22, 52
- Code, hexadecimal, 118
 - for characters, 46
- Colon:
 - in INPUT, 27
 - in PRINT, 31, 38
- Color, 221–36
- Commas in PRINT statements, 13–14, 16
- Components, 2
- Computer languages, 8
- Computer song, 82
- Concatentation, 123
- Condition (in IF-THEN STATEMENT), 42
- Controlling keyboard, 76–80
- Countdown to launch, 25
- CPU, 8
- Cursor, 4, 6
- Cursor moving keys, 9

D

- Dancing stick figure, 134–37
- DEL, 9

Deleting:

- character, 9
- program, 8
- Diagonal printing, 19
- Dice, 70
- Digits, random, 74
- Disk drive, 3
- Diskette, 3
- Distorting pictures, 49
- Doctor's assistant, 41
- Dollar sign, 27, 30, 33
- DONE, 13
- Doomsday machine, 212–13
- Dots on keys, 6
- Duration (in sound), 81

E

- Easter bunny, 36
- Easter card, 35
- EDIT, 10–13, 42
- END, 101
- Ending a line, 13
- Entering a program, 8
- ENTER key, 6, 8, 13, 25
- ERASE, 10
- Erasing line, 10
- Erasing old pictures, 53, 56
- Exploding star, 229–35
- Explosion, 106, 174–75
- Eyes in drawings, 66

F

- Factory, 167–68
- FCTN key, 5, 7, 9–10, 42
- File names (standard), 18
- FOR-NEXT statement:
 - backward, 19, 53, 112
 - slowdown loop, 23, 26
 - STEP, 19, 53, 112
- Four-letter words, 75
- Funny faces, 68–69

G

Geography quiz, 43–44
GOSUB statement, 63
 convenience, 163
GOTO statement, 44, 56
Graphic characters, 4, 45, 117–41
Graphics (screen size), 4, 45
Greater than (>), 149
Grid, 4
Ground (in drawings), 60, 100

H

Halloween card, 34
Handball, 60
Hardware, 2
Heart, 31
Hexadecimal code, 118
Highway, 113
HIT (message), 180–81, 219–20
Hyperspace warp, 211

I

IF-THEN statement, 42, 102
INCORRECT STATEMENT, 6
Index, 14
INPUT statement, 27–44
INS, 9
Inserting characters, 9
Interfaces, 3
INT statement, 70–75

J

Jet fighter, 176–81

K

Keyboard, 5, 76–80
Keyboard control of programs, 76–80

L

Language of computer, 8
Laser blast, 187–88, 210
Legal notice, 38

LEN, 124
Length of alphabetic string, 124
Less than (<), 149
Letters, random, 74–75
Line numbering, automatic, 24–25
Line numbers, 8
LIST, 9, 11
 variations, 18, 26
Listing a program, 9, 11
Living art, 236
LOAD (*see* OLD command)
Loading a program, 18
Loop, 14–15
Loop variable, 14
Lowercase letters, 8, 11
 random, 74

M

Making faces, 68–69
Mars warship, 26, 83
Memory, 3
Message board, 39
Meteorites, 217
Missile, 61, 193
Mistakes, correcting, 9
Modules, 144, 147
Momentum, 163–65
Mother's Day card, 35
Motion picture, 134
Movement:
 anywhere on screen, 152–65
 diagonally, 108–9
 downward, 87–93
 right-left, 102–4
 simultaneous, 189–96
Multicolors, 225
Multiplication, 70

N

Naming programs, 18
NEW command, 8, 11, 13, 28
New Year's card, 32
NEXT statement, 15
Noise, 81–86
Not equal to (< >), 101

NUM, 24–25, 28
Numbering lines (automatic), 24–25
Number mode, 24
Numbers, 74

O

Ocean waves, 191–92
OLD command, 18
On/off, 2, 3

P

Photograph of TI, 2
Picture of TI, 2
Pinchbug, 128–33
Pitch, in sound, 81
Planetary rover game, 214–20
Plastic strip, 6
Printer, 3, 5
PRINT statement, 12
 commas, 13–16
 quotation marks, 12–13, 40
 semicolons, 13–14, 17, 30
 to print a blank line, 31
Program, 8
PROGRAM CC-NN, 18
Program stub, 145, 209
Program writing methods, 72, 145,
 162–63, 175
Prompt, 27
Pulsating object, 112
Pumpkin, 34

Q

Questions, asking, 27–44
QUIT, 10
Quizzes, 42–44
Quotation marks, 12–13, 40

R

Rabbit, 36
Race car, 92–94, 126–27
Race car game, 197–206
Race track, drawing, 197, 204–5
Random art, 236

RANDOMIZE, 211
Random numbers, 70–75, 181, 217
Random placement on screen, 74
Random targets, 210–11
REM (remark) statement, 18, 48, 68
Repeating keys, 7
RES, 56, 65
Return key, (*see* ENTER key)
RETURN statement, 63
RND (random number), 70–75, 181,
 217
Rocket, 22–25, 57
 graphics, 124–25
 launch, 23
Rolling waves, 192
Roulette wheel, 70
RUN, 9, 13

S

SAVE, 17–18
Saving a program, 17–18
Schoolhouse, 227
Screen:
 areas, 14
 grid, 4
 locations for graphics, 4, 45, 70, 89
 saver, 5
 width for characters, 37
 width for graphics, 45
Screen areas, 14
Scrolling screen, 13
Semicolon, 13–14, 17, 30
Shadow elimination, 65, 87–88
Shadows in pictures, 65, 88
Shell, 169–72
SHIFT key, 5, 28
Ship drawing, 189
Simultaneous movement, 189–96
Slowdown:
 FOR-NEXT, 23, 26
 motion, 23
 using delays, 23, 26, 64
 using sound, 84–85
Slowdown, speedup, 74
Snowfall, 71

- Song, computer, 82
- Sound:
 - adding to program, 83
 - random, 82
 - slowdown, 85
- Sound effects, 137
 - explosion, 106, 174–75
 - rocket, 83
- Spacecraft, 86–98
- Spaces around picture, 71
- Spaceship, 105, 142–51
- Spaceship game, 148–50
- Space station, 61, 77–78
- Special characters (*see* Characters, custom)
- Speed control, 148, 175
- Speed object up, 74, 218
- Spider drawing, 88–89
- Standard name for programs, 18
- Star, 73, 229–35
- Starship, 154–58, 207–8
- Starship game, 207–13
- STEP (in FOR-NEXT statement), 19, 53, 112
- Stick figure, 60
 - dancing, 134–37
 - graphics, 119, 123
- Stopping the computer, 16, 76
- Stories, 37
- String, alphabetic, 27, 39
- String variable, 27, 30, 33
- Strip, plastic, 6
- Stub, program, 145, 209
- Submarine drawing, 190
- Subroutine, 63, 147

T

- TAB statement, 19
- Tank, 57–58, 183–85

- Target practice, 58, 78, 166–81
- Targets, 58, 100, 162, 212
 - factory, 78, 167–68
 - moving up and down, 178–81, 186
 - random location, 210–11
- Tax collector, 41
- Television display, 2, 4
- Top-down design, 145
- Torg dancing, 139–41
- Trampoline analogy (to FOR-NEXT statement), 15

U

- Uppercase characters, 10

V

- Valentine heart, 31
- Velocity, 175
- Video display, 2
- Volume, in sound, 81–82

W

- Walking bug, 63–65
- WARNING (message), 40
- Warp (hyperspace), 211
- Wave, ocean, 192
- White noise, 83
- Words, 75

X

- X, giant, 21

Animation, Games, and Sound For the TI 99/4A

Tony Fabbri

EXTRA! EXTRA! READ ALL ABOUT IT! . . . BUG WALKING RIGHT. . .
ROCKET LAUNCH WITH NOISE . . . TORG AND HIS TERRIBLE DANCE. . .
STARSHIP ATTACKING TARGETS INCLUDING DOOMSDAY TRIGGER. . .

These are examples of the dozens of fascinating programs featured in **Tony Fabbri's** new book. This book will show you how to draw pictures, make them move and walk, and create game action with keyboard control, moving targets, noise, and explosions. The author guides you step by step through every aspect of simple arcade games. Even if you have no background in computers or programming, you will soon be making sailboats and tanks move, launching rockets, drawing insects and alien creatures with blinking eyes, creating a game of catch, producing your own greeting cards, and devising your own colorful action games.

Some special features:

- includes many complete programs that you can use to create new games and handle practical applications;
- requires no equipment beyond the popular TI 99/4A computer;
- teaches the programming skills you need to solve business, educational, engineering, management, and scientific problems;
- lets you do things and see the results rather than just read about concepts or methods;
- shows you how to use the computer creatively and imaginatively.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-037227-7